International
Virtual
Observatory
Alliance

Units in the VO
Version 1.0

IVOA Proposed Recommendation 1.0-20131025

This version:
http://www.ivoa.net/Documents/VOUnits/20131025/

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

Previous versions:
http://www.ivoa.net/Documents/VOUnits/20131011/

Editor(s):
Sébastien Derrière and Norman Gray

Authors:
Markus Demleitner
Sébastien Derrière
Norman Gray
Mireille Louys
François Ochsenbein

Contents

List of Tables

Abstract

This document describes a recommended syntax for writing the string representation of unit labels (‘VOUnits’). In addition, it describes a set of recognised and deprecated units, which is as far as possible consistent with other relevant standards (BIPM, ISO/IEC and the IAU).

The intention is that units written to conform to this specification will likely also be parsable by other well-known parsers. To this end, we include machine-readable grammars for other units syntaxes.

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/`.

Note on conformance

Text within the following document is classified as either ‘normative’ or ‘informative’.

Normative text means information that is required to implement the Recommendation; an implementation of this Recommendation is conformant if it abides by all the prescriptions contained in normative text. **Informative** text is information provided to clarify or illustrate a requirement but which is not required for conformance.

The sections and subsections of this Recommendation are labeled, after the section heading, to specify whether they are normative or informative. If a subsection is not labeled, it has the same normativity as its parent section. References are normative if they are referred to within normative text.

When found within normative sections, the key words **must**, **must not**, **required**, **shall**, **shall not**, **should**, **should not**, **recommended**, **may**, **optional**, thus formatted, are to be interpreted as described in Bradner (1997).

Acknowledgements

We thank all those participants in IVOA and EuroVO workshops who have contributed by exposing use cases and providing comments, especially Paddy Leahy, Jeff Lusted, Jonathan McDowell, Marco Molinaro, Pedro Osuna, Anita Richards, Bruno Rino, Arnold Rots, Jesus Salgado Mark Taylor, Brian Thomas and recent contributors on the DM and Semantics forums.

1 Introduction (informative)

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 as far as possible 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. 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. 2.4. It is not, however, quite possible for VOUnits to be in the intersection of existing syntaxes; there is futher discussion of this point in Sect. 2.12.1.

We also provide, for information, a set of self- and mutually-consistent machine-readable grammars for all of the syntaxes discussed.

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 for the four syntaxes discussed.

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

1.1 Units in the VO Architecture

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

Units lie at the core of the VO architecture, as can be seen in Fig. 1. Most of the existing data and metadata collections accessible in the resource layer 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`

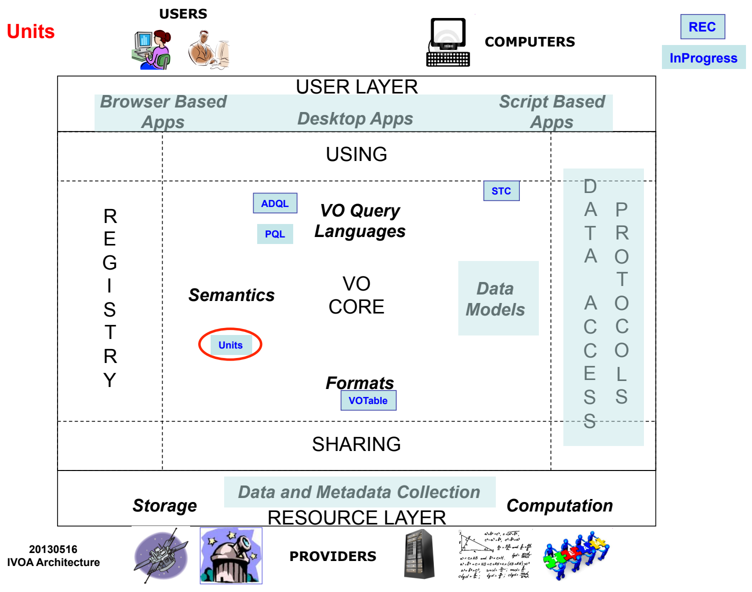


Figure 1: Units is a core building block in the VO. Most parts of the architecture rely on it: the User Layer with tools and clients, the Resource Layer with data. Protocols, registries entries, and data models also re-use these Units definitions.

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 terms of a given *unit*; there may be other structure to a quantity, such as uncertainty or even provenance. 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., 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 **unit** consists of a sequence of **unit components**, each of which represents a **base unit**, possibly modified by a multiplicative **prefix** (of one or two characters), and raised to an integer or rational power. The whole unit may (in some syntaxes) be prefixed by a numerical **scale-factor**.

Each of the **base units** (for example, the metre) is represented by a **base symbol** (for example `m`). Each syntax has a number of **known units** (Sect. 2.4), for each one of which there is at least one symbol which identifies only that unit.

A **symbol** is either a base symbol or a base symbol with a scaling string prefix.

For example, in the unit of `1.663e-1mm.s**-1`, the scalefactor is 1.663×10^{-1}, the two unit-components are `mm` and `s**-1`; the first symbol has base symbol `m` and prefix `m` (for 'milli'), and the second has base symbol `s`, no prefix, and the power -1.

1.3 Purpose of this document

The purpose of this document is to provide a reference specification of how to write VOUnits, in order to maximize interoperability within the VO; the intention is that VOUnit strings should be reliably parseable by humans *and* computers, with a single interpretation. This is broadly the case for the other existing unit-string syntaxes, although there are some slight ambiguities in the specifications of these syntaxes (cf Appx. C). We therefore include a set of self- and mutually-consistent machine-readable grammars for all of the syntaxes discussed.

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

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.

- In addition to the unit syntaxes described above, there are multiple specifications of base and known units (we refer, in particular, to specifications from BIPM, ISO/IEC and the IAU); these are broadly, but not completely, mutually consistent.
- Where there are some ambiguities in, or contradictions between, these various 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 existing syntaxes, and which we recommend that applications should use when writing unit strings. This aim is not quite possible in fact, and the extensions to it, 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 2 on page 13.

1.4 What this document will not do

This Recommendation does **not** prescribe what units data providers employ, except to the extent that we avoid giving a standard interpretation for a unit in some cases (for example we do not acknowledge the degree celsius or the century as units). Since we do not forbid 'unrecognised' units, this need not restrict data providers. Nor do we demand that a given quantity be expressed in a unique way (e.g., all distances in `m`). So long as data is 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. In particular, the Recommendation does not require that only recognised units are used. While it is obviously desirable for data providers to use recognised and non-deprecated units where possible, there are occasions when this is unnecessary or undesirable.

This Recommendation does not discuss *quantities* at all. That is, we do not discuss the combination of number and unit which refers to a particular physical measurement, such as '$2\,\mathrm{m\,s^{-1}}$'. Though this might appear to be a trivial extension, it raises questions of the representation of decimal numbers, the representation of uncertainties, questions of unit conversion, and other data-modelling imponderables which have in the past, possibly surprisingly, generated a great deal of discussion within the IVOA without, so far, a generally acceptable resolution.

This Recommendation describes only isolated units, and not arrays, records or other combinations of units. Several VO protocols require embedding complex objects into result tables, 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 required in such instances, their syntax will conform to, or include, this specification by reference.

In general, this Recommendation is concerned almost exclusively with the syntactic question of what is and is not a valid unit string, leaving most questions of interpretation to a higher layer in an application stack. Specifically:

- The specification does not forbid 'unknown' units. An implementation of this specification should be able to recognise, and communicate, that a unit is unknown, but it is not required to reject a unit string on the grounds that it is unrecognised.
- Similarly, although Table 2 on page 13 forbids some units from having SI prefixes, a VOUnit implementation should not itself reject a unit string which incorrectly includes a prefix, but should instead just make available the information that this has been detected, and that it is deprecated.
- The list of known units in Sect. 2.4 is not specific about the precise definitions of the units in question; for example, it refers to the 'second' without distinguishing between the various possible definitions that the second may have. In a particular context, a data provider may need to indicate which of a number of possible definitions is being used in fact. That said, a VOUnits processor must interpret the symbols of Table 2 on page 13 compatibly with the indicated units: a `m` is always a metre of one type or another, and may not be interpreted as, for example, a minute.

2 The VOUnits syntax (normative)

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 the SI standard (BIPM, 2006), except where noted.

2.1 String representation and encoding

VOUnits may occur in legacy contexts, in which the presence of non-ASCII characters may cause considerable technical inconvenience (for example FITS cards). There are only a few non-ASCII characters which we might wish to include in unit strings (for example Å or μ), and we can find substitutes for these sufficiently easily, that we feel there is little real benefit in permitting non-ASCII characters in VOUnit strings.

All the VOUnit characters in the specification below are printable ASCII characters (that is, in the range hexadecimal 20 to 7E); any extensions to this standard **should** be restricted to this same range.

All VOUnit strings **must** be regarded as case-sensitive (the strings in the other syntaxes are also case-sensitive).

2.2 Identifying prefixes within strings

A **symbol** within a unit-component **should** be parsed as follows:

1. If it corresponds to a known **base symbol**, then it **must** be recognised as such.
2. If the symbol starts with a multiplicative prefix, then this is recognised independently of whether the resulting base symbol is a known or unknown unit – thus `Mm` and `Mfurlong` are parsed as millions of metres and furlongs, but note that this implies, for the sake of consistency, that `furlong` is parsed as the femto-'urlong'.
3. In the VOUnits syntax (a significant divergence from the other syntaxes), base symbols **may** be put between single quotes `'...'` (ASCII character 27_{16}). Such symbols **must** be parsed as unrecognised unit symbols which are not further examined. See Sect. 2.11 for discussion.

2.3 Base units

There is good agreement for the base symbols across the different schemes (see Table 10 on page 26).

The VOUnits base symbols are listed in Table 1

`m`	(metre)	`g`	(gram)	`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 1: VOUnits base units

For masses, the SI unit is `kg`. However, existing specifications recommend not using scale factors with `kg`, but attaching them only to `g` instead.

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 Known units

In Table 2 on the following page, we indicate the 'known units' for each of the described syntaxes, which go beyond the physically motivated set of base units. 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.

Unrecognised units ***should*** *be accepted by parsers*, as long as they are parsed giving preference to the syntaxes and prefixes described here. Thus, for example, the string `furlong/week` **should** parse successfully (though perhaps with suitably prominent warnings) as the femto-'urlong' per week.

The Unity library (Sect. 3.2) recognises units with respect to a subset of the QUDT unit framework Hodgson et al. (2013), with some astronomy-specific additions. This is a particularly comprehensive collection of units, and we commend it to the IVOA community as a *lingua franca* for this type of work.

Future versions of this specification may add to the set of known units.

2.5 Binary units

The symbol 'b' is sometimes used for 'bits', but this is the SI symbol for 'barn', and this Recommendation aligns with the SI standard in this respect. Since the same symbol is sometimes used for 'bytes', it is probably best avoided in any case.

ISO/IEC 80000-13, item 13-9.c notes that the term 'byte' 'has been used for numbers of bits other than eight' in the past, but that it should now always be used for eight-bit bytes; we recommend the same interpretation here. The same source notes the theoretical confusion between the symbol `B` for 'byte' and for 'Bel'. We believe it would be perverse in our present context to recommend against using 'B' for byte, and so resolve this here in favour of 'byte' by mandating that `B` **must** be parsed as indicating the 'byte', that the `dB` is an unprefixable special-case unit (as discussed below), and by implication that the 'dB' **must not** be interpreted as a tenth of a byte.[1]

[1]We have no evidence that this has been a common source of confusion within the IVOA, or indeed anywhere else.

unit	*description*	*fits*	*ogip*	*cds*	*vou*
%	percent			·	
A	ampere	s	s	s	s
a	julian year	s		s	s
adu	ADU	·			s
Angstrom	angstrom	d		·	dp
angstrom	angstrom		·		d
arcmin	arc minute	·	·	·	s
arcsec	arc second	·	·	s	s
AU	astronomical unit	·	·	·	p
au	astronomical unit				·
Ba	besselian year	d			
barn	barn	sd	·	s	sd
beam	beam	·			s
bin	bin	·	·		s
bit	bit	s		s	sb
byte	byte	sp	·	s	sbp
B	byte				sb
C	coulomb	s	s	s	s
cd	candela	s	s	s	s
chan	channel	·	·		s
count	number	·	·		sp
Crab	crab		s		
ct	number	·		·	s
cy	julian century	·			
d	day	·	·	·	s
dB	decibel				·
D	debye	·		·	s
deg	degree (angle)	·	·	·	s
erg	erg	d	·		sd
eV	electron volt	s	s	s	s
F	farad	s	s	s	s
g	gramme	s	s	s	s
G	gauss	sd	·		sd
H	henry	s	s	s	s
h	hour	·	·	·	s
Hz	hertz	s	s	s	s
J	joule	s	s	s	s
Jy	jansky	s	s	s	s
K	kelvin	s	s	s	s
lm	lumen	s	s	s	s
lx	lux	s	s	s	s
lyr	light year	·	·		s
m	meter	s	s	s	s
mag	magnitudes	s	·	s	s
mas	milliarcsecond	·		·	·
min	minute (time)	·	·	·	s
mol	mole	s	s	s	s
N	newton	s	s	s	s
Ohm	ohm	s		s	s
ohm	ohm		s		
Pa	pascal	s	s	s	s
pc	parsec	s	s	s	s
ph	photon	·			s
photon	photon	p	·		sp
pix	pixel	·		·	s
pixel	pixel	p	·		sp
R	rayleigh	s			s
rad	radian	s	s	s	s
Ry	rydberg	·		s	s
s	second (time)	s	s	s	s
S	siemens	s	s	s	s
solLum	luminosity	·		·	s
solMass	solar mass	·		·	s
solRad	solar radius	·		·	s
sr	steradian	s	s	s	s
T	tesla	s	s	s	s
ta	year tropical	d			
u	AMU	·			s
V	volt	s	s	s	s
voxel	voxel	·	·		s
W	watt	s	s	s	s
Wb	weber	s	s	s	s
yr	julian year	sp	·	sp	sp

Table 2: Known units in the various 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, a 'b' that it is permitted to have binary prefixes, and a 'd' that it is recognised but deprecated. For those units which have alternative symbols, a 'p' indicates the preferred one.

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

`Ki`	kibi, 2^{10}
`Mi`	mebi, 2^{20}
`Gi`	gibi, 2^{30}
`Ti`	tebi, 2^{40}
`Pi`	pebi, 2^{50}
`Ei`	exbi, 2^{60}
`Zi`	zebi, 2^{70}
`Yi`	yobi, 2^{80}

Table 3: VOUnits prefixes: (a, left) decimal prefixes; (b, right) binary prefixes

2.6 Scale factors

Units **may** be prefixed by any of the 20 SI scale factors, or the eight binary scale factors. The SI scale factors – provided in Table 3a – are the same as those of BIPM (2006), of ISO/IEC 80000-1, §6.5.4, and of Pence et al. (2010, Table 5) (see also Table 11 on page 27 for further comparisons).

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

The SI prefixes of Table 3a ***must** always refer to multiples of 1000*, even when applied to binary units such as bit or byte; this follows the stipulations (and clarifying note) of BIPM (2006, §3.1), and the proscription of ISO/IEC 80000-1, §6.5.4. If data providers wish to use multiples of 1024 (ie, 2^{10}) for units such as bytes or bits, they **must** use the the binary prefixes of ISO/IEC 80000-13, §4, reproduced in Table 3b (these were originally specified in IEEE 1541).

Note: 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.

2.7 Astronomy symbols

Table 12 on page 28 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.

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.[2] The

[2]We therefore rule out interpreting dB/cd as 0.9 mbit/s.

`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

year **may** be expressed by `yr` (common practice), or `a`, as recommended by ISO (ISO/IEC 80000-3, Annex C) and the IAU (IAU Commission 5, 1989, Table 6). However peta-year must only be written `Pyr`, to avoid the collision with the pascal, `Pa`.

There are no VOUnit symbols for degrees celsius or century. Temperatures are expressed in kelvin (`K`), and a century corresponds to `ha` or `hyr`. Note that *this is a mild deviation from the SI standard*, which states that the 'hectare', with unit symbol `ha`, is a 'non-SI unit accepted for use' as a measure of land area (BIPM, 2006, table 6), and which acknowledges neither 'a' nor 'yr' as a symbol for year.[3]

The astronomical unit **should** be expressed in upper-case, `AU`, in order to follow legacy practice. It may also be written `au`, in the VOUnits syntax, on the ground that it would be perverse to prefer the atto-atomic-mass to the astronomical unit, in an astronomical unit specification. *This is a deviation* from the SI recommendation of 'ua' (BIPM, 2006, Table 7), but conformant with the IAU's recommendation of 'au' (IAU Division I, 2012).[4]

Because of the near-degeneracy between the decimal prefixes `d` and `da`, there is an ambiguity when parsing the unit `dadu` – is this the deka-`du` or the deci-`adu`? The only cases where this ambiguity is possible are those involving known units starting with 'a' (`da` is unambiguously a deci-year for the same reason that `d` is unambiguously a day, because the presence of a bare unit would be ungrammatical). We can think of no cases where the ambiguity is plausible enough that resolving it is worth the specification effort, so we deem the parse of `da.*` to be **unspecified**.

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

[3]If large telescope arrays feel they must talk of attojoules per hectare per century, for some reason, they're going to have to be careful how they do so; it's probably best not to even think about atto-Henrys.

[4]If you feel a burning desire to write about micro-years or atto atomic-mass, this document is not the place you need to look for help.

VOUnit	Correct interpretation	Incorrect
`Pa`	pascal	peta-year
`ha`	hecto-year	hectare
`cd`	candela	centi-day
`dB`	decibel	deci-byte
`B`	byte	bel
`au`	astronomical unit	atto-atomic-mass

Table 5: Possibly ambiguous units

2.8 Other symbols

Table 13 on page 29 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 10, 11 and 12.

However, in order to be compatible with legacy metadata, VOUnit parsers **should** be able to interpret symbols `angstrom` or `Angstrom` (for ångström), `barn`, `erg` and `G` (for gauss).

Table 14 on page 30 compares other miscellaneous symbols. The last set of VOUnits symbols, derived from this comparison, is in 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; see note)

Table 6: Miscellaneous VOUnits. Note: The question mark (`?`) **should** be reserved for cases when the unit is unknown. It is not, however, part of the list of known units or the VOUnits grammar, and should be checked for before unit parsing.

It can be noted that some of the units listed in Table 14 on page 30 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

`str1.str2`	Multiplication
`str1/str2`	Division
`str1**expr`	Raised to the power expr
`fn(str1)`	Function applied to a unit string

Table 7: Combination rules and mathematical expressions for VOUnits. See Appx. C.4 for the complete grammar.

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, for the sake of legacy metadata. The VOUnits symbol for magnitudes is `mag`. Note that all symbols like `count`, `photon`, `pixel` are always used in lower case and singular form.

The decibel, `dB` is listed in the SI specification (BIPM, 2006, Table 8) amongst a set of 'other non-SI units', and mentioned by ISO/IEC 80000-3, §0.5 in a 'Remark on logarithmic quantities'. The `dB` **must** be parsed as a unit by itself – as opposed to being parsed as the prefix 'd' qualifying the unit 'Bel' – and both the decibel and Bel **must not** be used with other scaling prefixes.

If there is no unit associated with a quantity (for example a quantity that is a character string, or unitless), data providers **should** indicate this with an empty string rather than blanks or dashes.

2.9 Mathematical expressions containing symbols

Table 15 on page 31 summarizes how, in the various existing syntaxes, mathematical operations may be applied on unit symbols for exponentiation, multiplication, division, and other computations.

The combination rules are where the largest discrepancies between the different schemes appear. The FITS document discusses the problem of trying to best accommodate the existing schemes (Pence et al., 2010, §4.3.1), without really resolving the problem. This and other ambiguities are discussed in the detailed syntaxes of Appx. C.

VOUnits follow a subset of the FITS rules, as summarized in Table 7.

As illustrated in Table 7, units may include a limited set of functional dependencies on other units. The set of functions recognised within VOUnits is the same as the set recommended by FITS, and listed in Table 8. As with unrecognised units, *parsers* ***should*** *accept unrecognised functions without error*, even if they deprecate them at some later processing stage. As described

`log(str1)`	Common Logarithm (to base 10)
`ln(str1)`	Natural Logarithm
`exp(str1)`	Exponential (e^{str1})
`sqrt(str1)`	Square root

Table 8: Functions of units.

in Sect. 2.11, functions may be quoted to indicate that they **must not** be interpreted as in this table. Note that since functions such as 'log' require dimensionless arguments, when a quantity x is (for example) represented by numbers labelled with units `log(Hz)`, that indicates that the numbers are related to x by the function $\log(x/(1\,\mathrm{Hz}))$.

2.10 The numerical scale-factor

A VOUnits unit string **may** start with a numerical scale-factor to indicate a derived unit. For example, the inch might appear as the unit of `25.4mm`. See Appx. C.4 for the syntax of the VOUnits numerical string.

A data provider may choose to use such a unit in order to represent a unit which is not listed as one of the VOUnit 'known units'. For example, given a VOTable column of masses relative to Jupiter's mass, one might label it as having units of `1.898E27kg` rather than `'jupiterMass'` (an 'unknown unit'). The *advantage* of doing so is that the data consumer can translate the column data into well-known physical units without further information, and the data source is thus self-contained. The *disadvantage* of doing so is (i) that the intention might be obscured (this is a type of provenance information); and (ii) that the measurements may be relative to the actual jupiter mass rather than merely expressed in those terms, so that they should change if the actual mass were to be refined as a result of a recalibration. The data provider retains the choice of which strategy to take.

This Recommendation does not prescribe how many significant figures should be in a scale-factor, nor whether it should be interpreted as single- or double-precision, nor how units with scale-factors should be compared for equality. All of these are implementation choices for the software which is handling the units.

2.11 Quoting unknown units

In the VOUnits syntax, base symbols may be put between single quotes `'...'` (a significant divergence from the other syntaxes). Such symbols **must** be parsed as unrecognised unit symbols which are not further examined.

This has two consequences. Firstly, it means that an unknown symbol which happens to start with an SI (or binary) prefix is not broken into a base symbol and prefix: thus `'furlong'` is parsed as expected, whereas `furlong` would be the femto-'urlong'. Secondly, a quoted symbol is parsed as an unrecognised unit, even if it would otherwise indicate a known unit; thus the unit `'m'` is parsed as an unknown unit 'm', and does not indicate the metre.

This facility means that a data provider may label data with units of, for example, `'martianDay'` or the `'B'`, while still remaining conformant with the VOUnits Recommendation, and without risking the leading `m` being misparsed as an SI prefix, or the 'B' being misparsed as a 'byte'.

Functions may also be quoted, with the same interpretation.

Quoted units can take prefixes (they are 'unknown units', so there are no restrictions on their usage), so that `m'furlong'` is a milli-furlong, and `m'm'` is a milli-'m'.

2.12 General rationale (informative)

2.12.1 Deviations from other syntaxes

The aspiration of the VOUnits work was that the syntax should be as much as possible in the intersection of the various pre-existing syntaxes, so that a unit string which conformed to the VOUnits syntax would be parseable in each of those other syntaxes. This has not been possible in fact, for four reasons.

1. The CDS syntax permits only a dot to indicate a product, and the OGIP syntax only a star, while FITS permits both. The VOUnits syntax uses a dot, so that non-trivial OGIP unit strings are therefore necessarily invalid VOUnits strings in this one respect.
2. The VOUnits syntax permits (but does not require) a scale-factor at the beginning of the string, which is not a power of 10. Only the CDS syntax permits a similar factor. See Sect. 2.10 for discussion.
3. Only the VOUnits syntax permits quoted units.
4. Only the VOUnits syntax permits the use of the binary prefixes of Table 3.

The first is both unavoidable in specification, and largely unavoidable in practice; the others are VOUnit extensions which a data provider may of course decline to take advantage of.

The scalefactor and quoted-units extensions are intended to support the case where the data provider wishes to distribute data including a unit which is 'unknown', but which the provider nonetheless feels is necessary or useful; this should be done only after weighing the considerations of Sects. 2.10 and 2.11. For the sake of consistency, and in order to allow constructions such as `M'jupiterMass'`, the grammar permits quoted units to take scaling prefixes; this is not often likely to be a good idea.

A VOUnits string which avoids the three extensions above will be parseable, with the same meaning, in the CDS and FITS syntaxes, and will be parseable by an OGIP parser if dots are replaced by stars.

2.12.2 Restrictions to ASCII

As described above, VOUnit unit strings are restricted to printable ASCII characters. While the two most prominent uses of these strings will be within VOTable attributes (`unit="..."`) and in XML serialisations of a data model (for example `<unit>...</unit>`), we also intend them to be usable within FITS files and within databases. Neither of the latter two contexts is necessarily unicode-friendly, so permitting non-ASCII characters in a unit string (such as Å or μ) is more likely than not to cause trouble.

Similarly, forbidding spaces within VOUnit strings removes one (minor) complication when recognising them in use.

2.12.3 Other units, and unit-like expressions

As noted above, the VOUnits syntax does not include structures such as arrays or tuples of numbers. We include in this category sexagesimal coordinates, calendar dates (in ISO-8601 form or otherwise), RA-Dec pairs, and other structured quantities serialised as strings. Each of these is well-specified elsewhere, and would require a separate parser if encountered in data.

Existing VO standards already recommend that coordinates be expressed in decimal degrees.

Quantities like the Modified Julian Date (MJD) are also 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 to overload VOUnits to incorporate the description of concepts themselves.

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

3 Use cases and applications (informative)

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[5], and can be tested online[6]. 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[7] is a new standalone library intended to parse unit strings in the VOUnits, OGIP, StdCats and FITS syntaxes; 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, such a library might be able to convert a distance in `pc` into a distance in `AU` or `km`, or convert 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 having a

[5] http://cds.u-strasbg.fr/resources/doku.php?id=units

[6] http://cdsweb.u-strasbg.fr/cgi-bin/Unit

[7] https://bitbucket.org/nxg/unity

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.
- 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

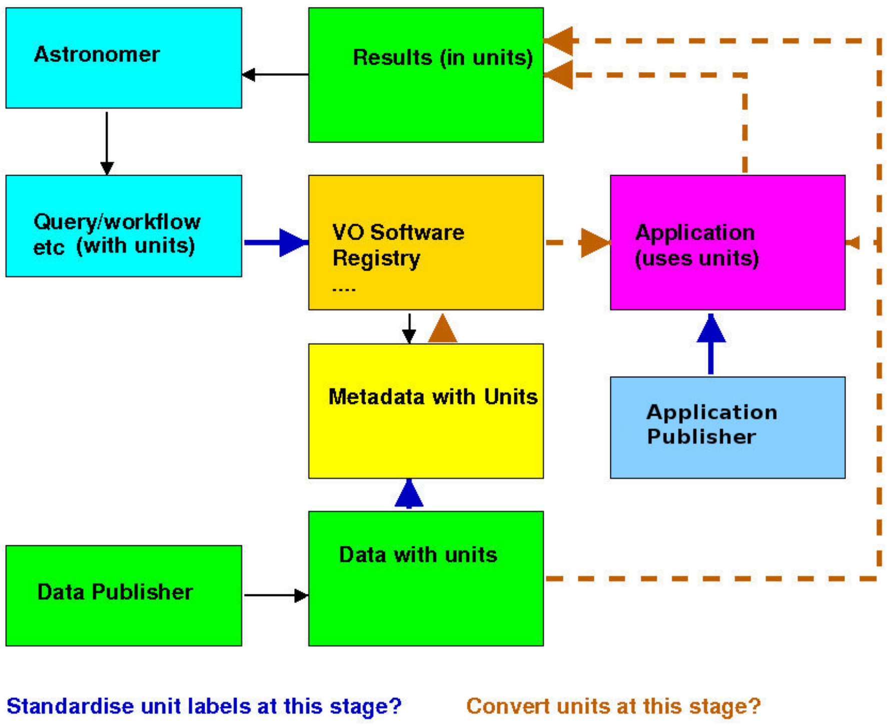
Including VOUnits in queries is not an easy task. Some guidelines were defined in the reflexion on ADQL.

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[8], 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:

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

(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



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

Figure 2: This shows the levels at which conversions might be done. Plain 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. Dashed arrows: Conversions required to supply results to the user in specified or user-prefered units e.g., `J.s-1` to `W`, are done where and when they are required.

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.

A Current use of units (informative)

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 History: Comparison of syntaxes (informative)

In this section, we compare the existing unit-string syntaxes and the proposed standard. We have included these comparisons for more-or-less historical reasons, to try to highlight the variations between syntaxes, and so illustrate the motivation motivation for this Recommendation, namely that the current practice, though it may at first appear to have rough consensus, is disturbingly heterogeneous.

	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 9: 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
	(1)	`kg`	`g`	`kg`, but `g` allowed	`g`
Dimensionless planar and solid angle	`rad, sr`				idem
				(2)	
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 10: Comparison of base units. Notes: (1) unit is `kg`, but use `g` with prefixes; (2) `deg` preferred for decimal angles

	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`				idem
	μ	`u` `z, y, Z, Y`			`u` `z, y, Z, Y`
Prefix–symbol concatenation	(1)	(2)	no space	no space (implicit)	no space
Prefix-able symbols	Not `kg`: use `g`	(3)	all	all	(4)
Use compound prefixes	should not	should never	must not	must not	must not

Table 11: Comparison of scale factors. Notes: (1) no space, regarded as single symbol; (2) no space, regarded as a single unit string; (3) all units above, and `eV, pc, Jy, Crab` Only `mCrab` allowed; (4) all (except `P` for `a`).

	IAU	OGIP	StdCats	FITS	VOUnits
minute	`min`, $^{\mathrm{m}}$	`min`	`min`	`min`	`min`
hour	`h`, $^{\mathrm{h}}$	`h`	`h`	`h`	`h`
day	`d`, $^{\mathrm{d}}$	`d`	`d`	`d`	`d`
year	`a`	`yr`	`a`, `yr`	`a`, `yr` (1)	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`		(2)
cycle	`c`, $^{\mathrm{c}}$				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	(3)				(4)

Table 12: Comparison of astronomy-related units. Notes: (1) Pa (peta-a) forbidden; (2) no dedicated symbol, use `uarcsec`; (3) ha, cy should not be used; (4) no dedicated symbol, use `ha` or `hyr`

	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`	(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 13: Comparison of symbols deprecated by IAU (from IAU Commission 5 (1989): "Table 7. Non-SI units and symbos whose continued use is deprecated"). Note: (1) no symbol – `mW/m2` used for $\mathrm{erg\,cm^{-2}\,s^{-1}}$.

	IAU	OGIP	StdCats	FITS	VOUnits
magnitude	mag				mag
rydberg			Ry	Ry	
solar mass	$M_\odot$		solMass	solMass	same as FITS
solar luminosity			solLum	solLum	
solar radius			solRad	solRad	
light year		lyr		lyr	
count		count	ct	ct, count	
photon		photon		photon, ph	
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 14: Comparison of other symbols.

	IAU	OGIP	StdCats	FITS
Multiplication	space or dot (1)	space or star (2)	dot	space or star (3)
Division	per (4)	`/` (5)	`/`, no space	`/`, no space
Use of multiple /	never	allowed	allowed	discour-aged (6)
`sym` raised to the power y	superscript	(7)	(8)	(9)
Exponential of `sym`		`exp(sym)`		`exp(sym)`
Natural log of `sym`		`ln(sym)`		`ln(sym)`
Decimal log of `sym`		`log(sym)`	`[sym]`	`log(sym)`
Square root of `sym`		`sqrt(sym)`		`sqrt(sym)`
Other math		(10)		not used
( )		allowed	allowed	optional around powers
powers	super-scripts	(11)	integers	(12)
Numeric factor	not used	(13)	allowed	(14)

Table 15: Mathematical expressions and symbol combinations. Notes: (1) space, except if previous unit ends with superscript; dot (`.`) may be used; (2) one or more spaces OR one asterisk (`*`) with optional spaces on either side; (3) single space OR asterisk (`*`, no spaces) OR dot (`.`, no spaces); (4) use negative index or solidus (`/`); (5) solidus (`/`) with optional spaces on either side, space not recommended after / OR negative index; (6) may be used, but discouraged, 'math precedence rule'; (7) `sym**(`y`)` parenthesis optional if $y > 0$; (8) nothing – `sym`y, and use $+/-$ sign for `10+21`; (9) `sym`y OR `sym**(`y`)` OR `sym^(`y`)`, no space; (10) f`(sym)`, where f is `sin`, `cos`, `tan`, `asin`, `acos`, `atan`, `sinh`, `cosh`, `tanh`; (11) decimal and integer fractions allowed; (12) integer (sign and () optional), OR decimal or ratio between (); (13) should be avoided; only powers of 10 allowed; should precede any unit string; (14) optional 10**k, 10^k, or 10±k.

C Formal grammars

Subsection C.4 is Normative, the other subsections are Informative.

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 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 from the 'Unity' package at `https://bitbucket.org/nxg/unity`, which includes machine-readable grammars, lists of recommended units, and a collection of test cases. These are also extracted in machine-readable form at `https://code.google.com/p/volute/source/browse/trunk/projects/std-vounits/unity-grammars.zip`.

In these grammars, the common terminals are as given in Table 16. Lexers **must not** swallow whitespace in generating these terminals; whitespace is permitted in a units string only where the corresponding grammar permits the `WHITESPACE` terminal.

`CARET`	the ^ character ($5E_{16}$)
`DIVISION`	the solidus, `/` ($2F_{16}$)
`DOT`	the dot/period/full-stop character ($2E_{16}$)
`FLOAT`	a string matching the regular expression `[-+]?[0-9]+\.[0-9]+`
`LIT10`	a literal string '`10`' (the sequence 31_{16} 30_{16})
`OPEN_P` / `CLOSE_P`	parentheses (28_{16} and 29_{16})
`SIGNED_INTEGER`	an integer with a required leading sign, so matching the regular expression `[-+][0-9]+`
`STAR`	the asterisk ($2A_{16}$)
`STARSTAR`	a pair of asterisks, `**`
`STRING`	a non-empty sequence of letters `[a-zA-Z]+`
`UNSIGNED_INTEGER`	an integer with no leading sign `[0-9]+`
`WHITESPACE`	a non-empty string of space characters (20_{16} only)

Table 16: The terminals used in the grammars; the notation NN_{16} indicates hexadecimal ASCII character numbers; the digits are 30_{16} to 39_{16}, the letters are 41_{16} to $5A_{16}$ and 61_{16} to $7A_{16}$, and the sign characters are $2B_{16}$ and $2D_{16}$.

C.1 The FITS grammar (informative)

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 17 on page 36.

As noted above in Sect. 2.9, 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 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.

The FITS specification permits a leading numeric multiplier, but "[c]reators of FITS files are encouraged to use the numeric multiplier only when the available standard scale factors of [SI] will not suffice".

The FITS specification permits `m(2)`, to indicate the square of unit 'm'. The grammar has to special-case this, in order to distinguish it from function application.

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 The OGIP grammar (informative)

For the OGIP units syntax, see George and Angelini (1995). Our preferred OGIP grammar is in Table 18 on page 37.

The OGIP specification somewhat reluctantly concedes (in its section 3.2) that "occasionally it may be preferable to include [leading scale] factors on the grounds of user-friendliness", but that "[t]he inclusion of numerical factors should therefore be avoided wherever possible", and it is "suggested" that the scale factor should in any case be restricted to powers of 10.

Specification ambiguities:

- The OGIP specification permits a space between the leading factor and the rest of the unit (by implication from the provided examples).
- The specification does not indicate the format of the numerical factor in the case where it is not a power of ten. We have suggested `FLOAT` here (see Table 16 on page 32).
- 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 (informative)

For the CDS units syntax, see (CDS, 2000, §3.2). Our preferred CDS grammar is in Table 19 on page 38. It requires additional terminals, described in Table 20 on page 38.

Specification ambiguities:

- The CDS document indicates that units should be raised to powers by concatenation of the unit string with an integer, but does so rather elliptically, so that it is not clear whether `m+2` is permitted (the relevant examples show this as `m2`). We take this to be permitted in this grammar.
- The specification does not indicate the format of the numerical factor in the case where it is not a power of ten and not a `CDSFLOAT`. We have suggested `FLOAT` here (see Table 16 on page 32).
- The document does not specify or illustrate how `kg/m/s` should be parsed. Since the document mentions the OGIP standard (even though it does not permit OGIP's syntax for powers, `m**2`), we take it that this is valid, and equivalent to $\mathrm{kg\,m^{-1}\,s^{-1}}$.

This specification places no restrictions on the leading scale factor.

C.4 The VOUnits grammar (normative)

The VOUnits grammar is defined by this section, by the grammar in Table 21 on page 39 (with the terminals of Table 16 on page 32 plus the extra ones listed in Table 22 on page 39) and by the list of known units of Table 2 on page 13.

The intention of the VOUnits grammar is that if a VOUnits string does not use the scalefactor, quoted-units or binary-prefix extensions (that is, if it avoids the `VOUFLOAT` and `QUOTED_STRING` terminals and is restricted to SI decimal prefixes), then it will be parseable, with the same semantics, by FITS and CDS parsers, and that it will be parseable by an OGIP parser if dots are replaced by stars. See Sect. 2.12.1 for discussion. 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.

In Table 21 on page 39, the `VOUFLOAT` terminal is a string matching either of the regular expressions

- `0\.[0-9]+([eE][+-]?[0-9]+)?`
- `[1-9][0-9]*(\.[0-9]+)?([eE][+-]?[0-9]+)?`

(that is, something resembling `0.123` or `1.5e+11`).

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

complete_expression: product_of_units
        | product_of_units division unit_expression

product_of_units: unit_expression
        | product_of_units product unit_expression

unit_expression: term
        // m(2) is m^2, not function application
        | STRING parenthesized_number
        | function_application
        | OPEN_P complete_expression CLOSE_P

function_application: STRING OPEN_P complete_expression CLOSE_P

scalefactor: LIT10 power numeric_power
        | LIT10 SIGNED_INTEGER

division: DIVISION

term: unit
        | unit numeric_power
        | unit power numeric_power

unit: STRING

power: CARET
        | STARSTAR

numeric_power: integer
        | parenthesized_number

parenthesized_number: 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 17: The FITS grammar. See Appx. C.1.

```
input: complete_expression
        | scalefactor complete_expression
        | scalefactor WHITESPACE complete_expression

complete_expression: 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: term
        | function_application
        | OPEN_P complete_expression CLOSE_P

function_application: STRING OPEN_P complete_expression CLOSE_P

scalefactor: LIT10 power numeric_power
        | LIT10
        | FLOAT

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

term: unit
        | unit power numeric_power

unit: STRING

power: STARSTAR

numeric_power: UNSIGNED_INTEGER
        | FLOAT
        | parenthesized_number

parenthesized_number: 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 18: The OGIP grammar. Note that the `FLOAT` in the `scalefactor` production must be a power of ten. See Appx. C.2.

```
input: complete_expression
        | scalefactor complete_expression

complete_expression: 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: term
        | function_application
        | OPEN_P complete_expression CLOSE_P

function_application: OPEN_SQ complete_expression CLOSE_SQ

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

division: DIVISION

term: unit
        | unit numeric_power

unit: STRING
        | PERCENT

power: STARSTAR

numeric_power: integer


integer: SIGNED_INTEGER | UNSIGNED_INTEGER

product: DOT
```

Table 19: The CDS grammar. See Appx. C.3 for discussion, and Table 20 for the additional terminals.

`CDSFLOAT`	a string matching the regular expression `[0-9]+\.[0-9]+x10[-+][0-9]+` (that is, something resembling `1.5x10+11`)
`OPEN_SQ`	the open square bracket '`[`' (indicates logs in this syntax)
`CLOSE_SQ`	the close square bracket '`]`'
`PERCENT`	the percent character '%'

Table 20: Extra terminals for the CDS grammar

```
input: complete_expression
        | scalefactor complete_expression

complete_expression: product_of_units
        | product_of_units division unit_expression

product_of_units: unit_expression
        | product_of_units product unit_expression

unit_expression: term
        | function_application
        | OPEN_P complete_expression CLOSE_P

function_application: STRING OPEN_P complete_expression CLOSE_P
        | QUOTED_STRING OPEN_P complete_expression CLOSE_P

scalefactor: LIT10 power numeric_power
        | LIT10
        | VOUFLOAT

division: DIVISION

term: unit
        | unit power numeric_power

unit: STRING
        | QUOTED_STRING
        | STRING QUOTED_STRING

power: STARSTAR

numeric_power: integer
        | parenthesized_number

parenthesized_number: 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 21: The VOUnits grammar. See Appx. C.4 for discussion, and Table 22 for additional terminals.

`VOUFLOAT`	see text
`QUOTED_STRING`	a `STRING` between single quote marks (ASCII character 27_{16})

Table 22: Extra terminals for the VOUnits grammar

D Updates of this document (informative)

- 1.0-20131025:
  - Grammar changes: The '%' character is now treated as a special case, rather than being a permitted 'STRING' character; it's only the CDS syntax that permits this character. Some readability adjustments to the grammars. Unit strings with leading slashes (eg `/m3`) are no longer supported in the VOUnits syntax. The grammars now match Unity v0.10.
  - Changed discussion/rationale for forbidding non-ASCII characters.
  - Clarified that '?' – which is specified as indicating an unknown unit – is not part of the VOUnits grammar, and should be spotted by a caller before parsing begins.
  - Clarified the extra terminals which some grammars use.
  - Clarified that the ambiguity in `dadu` should remain unresolved, and the correct behaviour unspecified (is it deci-`adu` or deka-`du`?).
- 1.0-20131011: Changed gramme in gram; removed color property to distinguish arrows in fig .2; Removed astro'l unit abbreviation from known-units.tex
- 1.0-20130922: Responding to RFC and mailing list comments. Addition of quoted units and arbitrary scale-factor (so updates to grammars, which now match Unity v0.9). Some reformatting of tables.
- 1.0-20130724: Rephrasing and clarification, responding to RFC comments. Update unity grammars to current version (ie, version of 2013-07-22 18:40).
- 1.0-20130701: Simplified Architecture diagram. Added example with scientific notation. Adjusted locations of grammar tables to try to keep them closer to the associated text.
- 1.0-20130429: Some restructuring, some rephrasing, and a few layout changes.
- 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.
- 1.0-20121212: 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.
- 1.0-20120801: Minor typographical fixes
- 1.0-20120801:
  - Included yacc-style grammars in document.

- 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.
- 1.0-20120521:
  - 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. 2.6.
  - Micro-arcsecond and century explained in Table 12 on page 28.
  - Table 13 on page 29 completed.
  - Added numeric factors in Table 15 on page 31 and discussion in text.
- 1.0-20111216: Major rework of the document.
- 0.3: initial public release.

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/.

S Bradner. Key words for use in RFCs to indicate requirement levels. RFC 2119, March 1997. URL http://www.ietf.org/rfc/rfc2119.txt.

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/.

Ralph Hodgson, Paul J. Keller, Jack Hodges, and Jack Spivak. QUDT: Quantities, units, dimensions and data types ontologies, 2013. URL http://qudt.org/.

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. See also http://www.iau.org/science/publications/proceedings_rules/units/.

IAU Division I. Resolution B2: on the re-definition of the astronomical unit of length, 2012. URL http://www.iau.org/static/resolutions/IAU2012_English.pdf.

IEEE 1541. IEEE standard for prefixes for binary multiples (IEEE 1541-2002). IEEE Standard, 2002. See also IEC-80000-13.

ISO/IEC 80000-1. Quantities and units — part 1: General (ISO 80000-3:2009). International Standard, 2009. Supersedes ISO 1000 and ISO 31-0.

ISO/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.

ISO/IEC 80000-3. Quantities and units — part 3: Space and time (ISO 80000-3:2007). International Standard, 2007. Supersedes ISO 31-1 and ISO 31-2.

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.