This is the revised version that I promised at the Interoperability Workshop. I have substantially changed the content and scope of this document in ways that go well beyond the issues discussed directly in the UCD working group meeting. These changes are reflected primarily in section 5. I have included some commentary on the text within the document. This text is generally given in red and would be eliminated even in the unlikely event that this proposal were accepted without change. The commentary material includes discussion of differences between this proposal and the previous one. There is a summary of this after section 4.

Metadata Content within VO Resources
Version 1.9.9b

IVOA Working Draft

2003-10-21

Previous versions: 1.9.9 (2003-10-02)

Authors:

Sébastien **Derriere** <derriere@newb6.u-strasbg.fr>
Norman **Gray** <norman@astro.gla.ac.uk>
Bob **Mann** <rgm@roe.ac.uk>
Andrea Preite **Martinez** <andrea@rm.iasf.cnr.it>
Jonathan **McDowell** <jcm@cfa.harvard.edu>
Thomas **McGlynn** <<Thomas.A.McGlynn@nasa.gov>
François **Ochsenbein** <francois@vizir.u-strasbg.fr>
Pedro **Osuna** <Pedro.Osuna@esa.int>
Guy **Rixon** <gtr@ast.cam.ac.uk>
Roy **Williams** <roy@cacr.caltech.edu>

1. Abstract

This document defines and describes the standard method for incorporating metadata content descriptors within Virtual Observatory entities. In includes both the definition of these Uniform Content Descriptors (UCDs) and specific recommendations for how these quantities should be used within tabular information. While use in tabular datasets is highlighted, UCDs may be used in other contexts.

2. Status of this document

This is an IVOA Proposed Recommendation for review by IVOA members and other interested parties. It is a draft document and may be updated, replaced, or superseded by other documents at any time. It is inappropriate to use IVOA Working Drafts as reference materials or to cite them as other than "work in progress." A list of current IVOA Recommendations and other technical documents can be found at http://www.ivoa.net/Documents/.

This document proposes a revision to an accepted UCD vocabulary. This revision is not compatible with the existing vocabulary. Acceptance of this document as a recommendation implies changes to existing software and other informal agreements including the Cone Search and Simple Image Access Protocols since these use UCDs incompatible with the proposed standard.

3. Acknowledgments

This document is based on the W3C documentation standards, but has been adapted for the IVOA.

4. Uniform Content Descriptors: A Controlled Vocabulary for Astronomy

The Unified Content Descriptor (UCD) is a formal vocabulary for astronomical metadata that is controlled by the International Virtual Observatory Alliance (IVOA). Each UCD describes a concept, a kind of thing that may be of interest to astronomers or useful to VO resources. UCD concepts are analogous to classes in object oriented programming. In particular the UCD describes the type of the thing, the content it refers to defines one or more instances of the given concept. E.g., Galactic latitude is a concept, the latitude of the Galactic center which has the numerical value of 0, is an instance of this concept. The UCD concept vocabulary is *restricted* in order to avoid proliferation of terms and synonyms, and *controlled* in order to reduce ambiguity as far as possible.

UCDs play a central role in resource discovery and processing in the Virtual Observatory. Services may publish the UCDs that describe their outputs. Looking at these UCDs human users or machine agents can find sources of the information they are looking for. When a VO service needs to process a element of data obtained from a remote site, the UCDs guide the processing. Existing and proposed VO protocols use UCDs as the mechanism to convey to users where critical information resides. UCDs and trees of UCDs may be used to express the relationships among data critical to building effective data models.

UCDs are simply a string with a structure defined below. The format of the string balances a number of competing goals:

1. UCDs should be short.
2. They should suggest the concept being labeled.
3. Only a single UCD should be appropriate for a given concept.
4. UCDs should be complete, describing all concepts of interest.

5. The vocabulary used within UCDs should be as small as possible.
6. The structure of UCDs should reflect the relationships of the concepts they label. Related concepts should have related UCDs.
7. Quantities with the same UCD should be comparable. This 'comparability' may requiring rescaling or adjustment of the values or attributes of the instances being compared. Comparability of UCDs is discussed in section 4.5, including the meaning of comparability for UCDs that describe complex entities.

A UCD description of a quantity does not define the units or name of the quantity, but rather *'what sort of quantity is this?'*; for example `phys.temperature` is a semantic class description of temperature, without implying a particular unit.

It would be possible to describe astronomical data quantities in a natural language such as English or Hungarian or Uzbek; however, it would be very difficult to expect a machine to 'understand' in any sense. At the opposite extreme, there is an attempt within the IVOA to describe astronomical data in terms of a hierarchical data model, so that there is a place for everything, and everything is in its place. The UCD vocabulary falls between these extremes, and is (we hope) understandable to both human and computer.

I have deleted the original section 3.1 since I don't believe it adds anything to this specification. If we plan a UCD 3, those goals might better be defined in a conclusion or appendix.

4,1 UCD Syntax

A UCD is a string which contains textual tokens that we shall call *words*, which are separated by semicolons. A word may be composed of several *atoms*, separated by period characters. The order of these atoms induces a hierarchy. E.g., the set of words pos, pos.eq and pos.eq.ra reflect increasingly precise concepts. Standard words used in the UCD, which are validated by the IVOA, can start with the `ivoa:` namespace, but this namespace is optional.

The character set that may be used in a UCD is the upper and lower-case alphabet, digits, and hyphen. The colon, semicolon, and period are special characters as discussed above.

- The UCD syntax is case-insensitive – all uppercase characters should be converted to lowercase before parsing.
- There should be no whitespace within a UCD.

Each UCD consists of a sequence of words where the first word is a base concept and following words are properties. Properties comprise two types: modifiers and attributes. Zero or more modifiers words immediately follow the base concept, and these are followed by one or more attribute words. Modifiers are UCD adjectives, constraining the meaning of the base concept. Attributes define an aspect of the concept. For many simple concepts the only attribute may be the 'value'. E.g., consider a column that simple contains a set of flux measurements. The underlying concept is phot.flux and the UCD is phot.flux;value. As a convenience to the user in the case where the UCD is a simple two-word form, *concept*;value the value property may be omitted. E.g., we may use phot.flux as an abbreviation for phot.flux;value. However in phot.flux;em.radio;value the value attribute is mandatory.

The vocabulary of words used to define UCDs consists of three elements: base concepts, modifying properties and attribute properties. Words belong to only one of these classes but the component atoms that comprise words may appear in any of the three vocabularies. Subsequent sections of this document describe each of these three elements in turn.

Refined or special concepts are built in steps from a simple UCD.

phot.flux	A flux in some undefined band (short for phot.flux;value)
phot.flux;em.optical;value	A flux measurement in the optical band.
phot.flux;em optical;meas.error .	The error in the flux. This illustrates a different attribute of the flux concept. The UCD phot.flux;em.optical;meas.error is also a concept in its own right and thus may be modified.
phot.flux;em.optical;meas.error;stat.max	Only attributes may be appended to complete UCDs. This one indicates a column or more likely a parameter that is the maximum error in the flux.

A well-formed UCD comprises a base property, 0 or more modifying properties, and one or attribute properties. If more than one modifier is included then modifiers should be given in alphabetical order. The constraint on the order of modifiers helps to ensure that UCDs are unique. E.g., consider the UCD for a calculated optical flux:

phot.flux;em.optical;intent.calculated;value

This represents the same concept as

phot.flux;intent.calculated;em.optical;value

By specifying the order of modifiers users are able to match UCDs by simple string comparisons. A key issue to remember is that modifiers do not modify each other, but the base concept.

On the other hand the order of attributes is very significant and unconstrained by the syntax. The error in the maximum is a very different thing than the maximum of the error.

The type of a given word in the UCD vocabulary can be inferred from the initial atom. Words beginning with em, frame or intent are modifiers. The initial atom stat or filter signals an attribute. The single atom words value, vector, local, instance and multiplet are special attributes. All other words are base concepts.

The need for well-formed UCDs (or some equivalent) is needed in the old proposal as well. If we have more than one modifier, then the modifier property (or concept, I'm not sure which is appropriate in the old proposal) then there doesn't seem to be any natural way to specify the order of the attributes. Since the old proposal allows a fairly arbitrary set of related concepts and properties in the UCD assessing the 'equality' of two UCDs would be extremely difficult.

However the requirement that the UCD end in an attribute property is required by this new scheme to ensure that regular expression matches of the type that Aladin currently does can be done with good reliability.

4.1.2 Namespaces.

The use of namespaces, indicated by the presence of a colon in the word is possible, but should be avoided as far as possible. The namespace is defined by the string before the colon and the word follows. The words in the non-standard namespace must be distinct from all words currently in the IVOA namespace.

While developers may need local namespace, they should be used only temporarily, for words that are not yet included into the UCD validated by the IVOA. New words should be added using the procedures discussed in section XX.

4.1.3 Examples of Legal Syntax

The following examples have legal UCD syntax:

1 `pos.eq.ra;value`
2 `pos.eq.ra;meas.error`
3 `ivoa:meta.id;value`
4 `meta.id;value`
5 `Meta.ID;Value`
6 `x1:experimental.quantity;x2:new.modifier;stat.error`

In this list, 3 ,4 and 5 are all equivalent becuase `ivoa:` is the default namespace and UCDs are case insensitive. Note that the namespace applies separately to each word in the UCD. Thus example 6 has words from two non-standard namespaces and one from the standard namespace.

4.1.4 Examples of Illegal Syntax

The following UCDs are invalid. The table indicates the reason.

1	`pos.eq.ra; value`	Embedded space
2	`pos.eq.ra;;meas.error`	Null word
3	`pos..eq.ra;meas.error`	Null atom
4	`phot.flux;x1:meas.error`	Namespace reuses existing word
5	`Meas.Error`	No base concept
6	`Math.ratio;Phot.flux;Time.exposure`	Base concepts used after first word. Grouping constructs (section 5) would be used to associate columns.
7	`Phot.flux;em.polarized`	Does not terminate in an attribute
8	`Phot.flux;intent.calculated;em.xray;meas.error`	Modifiers not in alphabetical order.
9	`Flux;meas.error;em.optical;stat.max`	Modifier appears after an attribute

4.1.5 Backus-Naur Form

```
<alpha>             ::=  a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z
                        |A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z
<digit>             ::=  0|1|2|3|4|5|6|7|8|9
<char>              ::=  <alpha>|<digit>|-
<period>            ::== .
<semicolon>         ::== ;
<colon>             ::== :
<word-component>    ::=  <alpha>|<digit>|<word-component><char>
<namespace-ref>     ::=  <word-component>
<word>              ::= <word-component>|<word> <period> <word-component>
<nword>             ::= <namespace-ref> <colon> <word> | <word>
<BaseConcept>       ::= <nword>
<Modifiers>         ::=
                        <nword>;<Modifiers>
<Attrbutes>         ::= <nword>
                        <nword>;<Attributes>
<UCD>               ::= <BaseConcept>      # Special case for defaulting value
                        <BaseConcept>;<Modifiers><Attributes>
```

A UCD is always case-insensitive

4.2 UCD Vocabulary

This and following subsections describe the three kinds of word used within UCDs. Words belong to only one of the three types. For each section the initial atom of appropriate trees and a few example words are given. While additional trees will doubtless be developed, re-use of the existing hierarchies is encouraged.

4.2.1 Base Concepts.

This section discusses base concepts one of which starts each UCD. .

arith quantities related to arithmetic and mathematics, including count, difference, ratio. See the discussion in the attribute filter. Use of the arith tree should be restricted to cases when there is no existing UCD that describes the kind of row created. E.g., for a table of X-ray sources with a counts and exposure column, the proper column for the ratio of these two columns is almost certainly in the phot.flux tree rather than arith.ratio.

concept This single word is used as the origin of the concept tree. While it is rarely needed to be used in UCDs it can be used to affirmatively declare that the table writer doesn't know or does not wish to publish any semantic information about this entity. This has much the same effect as not including the UCD attribute, but if one is looking for columns with undefined UCDs it may be easier to search for the particular string UCD=concept than to find the locations where UCD is omitted. It can also be used in UCD templates where it might be replaced by a base concept from any tree, or in situations where no single tree is appropriate.

human quantities related to people and institutions: names, addresses, phone numbers,nationalities.

`meta`	quantities related to metadata, such as IVO style identifiers, flags, notes, URL
`instr`	quantities related to an instrument; typical sub-levels are telescope, observatory, etc.
`obs`	observation methods such as detector, filter, plate, spectrograph, exposure time, etc.
`phot`	All photometric measurements, organized according to the wavelength; includes polarization. I'm a little confused by phrase 'according to wavelength' that I've retained above. Is the em breakdown which used to be included inside the phot stuff now moved entirely to the em hierarchy. I'm assuming that we use phot.flux;em.optical.u rather than phot.flux.optical, If so we need to change the above.
`phys`	Generic physical quantities, such as length, velocity, mass, and including atomic & molecular concepts and properties, temperature, pressure, gravity, etc...
`pos`	Position in the sky, reference frames; including equatorial, galactic etc coordinates; geocentric, heliocentric etc; and precession and nutation. Also includes position on the surface of the Earth.
`soft`	Software defined objects, e.g., services, archives, catalogs
`spec`	Quantities related to spectroscopic measurements
`src`	properties of the observed source of radiation: common ids, source classifications and morphology, extension in the sky, variability,
`time`	Quantities related to time.

The new tree was simplified compared to the old UCD tree; important modifications include:

- Atomic and molecular data are moved to a branch of `phys`
- The `src` branch is introduced for photon sources such as stars, galaxies etc.
- Observatory information is moved to the `instr` branch.
- A human tree is added to hold concepts relating to the people and institutions
- A soft tree is added to hold concepts relating to software entities that may be described in VO entities.
- The single word concept is added to be the root of all concepts.

More details about some of the most important branches are shown in the annex below.

4.2.2 Modifier properties.

These words are used to modify the meaning of a base concept. Changes and additions to the modifier tree should be made with somewhat greater care than for base concepts. Users should carefully consider how the modifier works in conjunction with the all base concepts. Modifiers should be carefully chosen so as not to introduce ambiguities in the formulation of UCDs.

em — quantities which describe the kind of electromagnetic radiation involved in the base concept, notable frequency/energy/wavelength and polarization. Since many of these same quantities apply to gravitational radiation and some to particle detections, the em atom may be interpreted loosely in these cases.

frame — quantities which describe the frame in which the base concept is to be taken, i.e., the context in which the base concept is realized. The em qualifier addresses the energy dimension of the frame and the spatial element is typically included in the base concept (cf. the eq and gal fields inside pos). However time frames should generally be addressed here. E.g., a UCD might be defined as time;frame.time.geocentric;value. The frame attribute might also be used modify spatial coordinate systems as in: pos.body.lat;frame.body.mars;value.

Almost all modifiers can be thought of as being in the frame tree, e.g., the em and intent trees could easily be rendered as frame.em and frame.human, but are separated out since they are so common.

intent — quantities that restrict the human context of the concept. E.g., calculated, predicted, simulated. This should not be used to distinguish among distinct science elements, e.g., background versus foreground which would normally be part of the base concept or the measurement if they are addressed at all.

The modifiers include two new trees, frame and intent which are discussed above.

4.2.3 Attribute properties.

Attribute properties indicate that the given UCD labels an attribute of the base concept. E.g., the value,meas.error,and stat.max attributes describe entities that give values, errors and maximum of the base concept. Note that the UCD indicates a semantic relationship and not necessarily a physical relationship. E.g., suppose a user submits a request rendered in SQL as "select max(ra) from table". This return from this query might be a single cell table with a UCD for the column as pos.eq.ra;stat.max even though there is no right ascension column present in the table the user sees.

This does not mean that attribute properties cannot be used to determine the likely relationships among columns in a table. How this is done is discussed in section 5.

As with modifiers attributes should be extended with considerable care. Additions to the single word attributes are should be particularly scrupulously considered.

The following attribute trees are defined.

filter — A filter is an attribute that reflects processing on some other column. The attribute trees for filter and stat and the concept tree for arith are linked but distinct.

Arith should normally be reserved for concepts that involve the combination of multiple columns so that there is no single base UCD. The association of an arith

column with the columns it derives from is discussed in section 5.

Stat attributes refer to processing of single columns that normally reduces the dimensionality of the data (e.g., a data column is transformed into some statistic on the column). There will normally be a single value of stat.max associated with all of the values in a column. Thus for simple VOTables the stat attributes may often be found in PARAMETER elements rather than FIELD elements. However if a column has vector valued cells, then there may be a stat.max attribute associated with each row – but it will be a scalar. Occasionally stat quantities may have the same dimensionality in the table as the base concept, but even this will often reflect the fact that the max column pointed to an array of data that has not been recorded. E.g., we might record the temperature of aninstrument at the beginning of each minute as well as the maximum and minimum temperatures during the minute. The value and stat.max columns have the same dimensionality, but the maximum was derived from a higher dimensionality dataset.

The filter tree reflects processing of a single column that preserves its dimensionality. Filters may be common for vector columns. Attributes would include: smooth, compress, rotate.

meas

Attributes associated with the process of measurement. The attribute meas.value should not be used (use just value instead). Error, error.statistical, error.systematic are key members of this tree.

See the discussion for filter to understand why error was taken out of the stat tree, i.e., error has the same dimensionality as value. It also doesn't really fit semantically into stat.

stat

Attributes that reflect the calculation of a statistic on a collection of elements described in the base concept. Where appropriate attributes like stat.max may be used in alphanumeric as well as numerical contexts, but many stat elements are useful only in numerical contexts. See the discussion in filter above.

The filter and meas trees are new.

There are a few new special single word attributes which we now discuss. These attributes play a central role in building complex UCDs.

value

The value attribute is the basic attribute for simple concepts. It represents the actual instantiation of the string or number that is implied by the concept. A value attribute should never be used after another attribute. I.e., phot.flux;meas.error;value.is incorrect: the value is superfluous. Note that if the basic concept is vector-valued, the value may be a vector. I.e., consider cell that contains an array of fluxes. Since UCDs are intended to convey the semantic content of the cell, it would be appropriate to distinguish whether the information in the cell was a time series or a spectrum. If we define appropriate UCDs phot.spectrum;em.optical;value and phot.timeseries;em.optical;value we can distinguish these two situations.

vector

This attribute may be used when there is no special UCD describing an array of the base concept. While VOTables provide a dimension attribute, it seems likely that users will find being able to distinguish vector and scalar columns in the UCD

useful. Errors and filters associated with vector quantities would also likely be vectors. Note that some UCDs should imply a vector column. E.g., a phot.spectrum or phot.timeseries might be used to describe a cell with an array of flux values. The UCD phot.spectrum;vector would imply a vector of spectra in each cell.

`instance`

This play the role of value for complex UCDs, i.e., the UCDs of tables and groups. It indicates that all of the fields and parameters in the appropriate scope are attributes of the concept described in this UCD. E.g., a simple table UCD might be src;instance. The instance indicates that in each row the ra, dec, flux, name, etc. refer to a single particular source.

`multiplet`

This can be used instead of instance in complex UCDs to indicate that there are several of the same instances in each row of the table. This attribute should rarely be used, but might be appropriate in a context where two tables have been cross-correlated and neither table is considered subordinate to the other. E.g., after cross-correlating the ROSAT and ASCA observations over the sky, the result is a table each row of which represents a pair of observations.

`local`

This special attribute asserts that the semantics of the column are such that it cannot be used outside of its local context. In particular it indicates that this column should not be cross-correlated with any other columns.

Note that local should be used when the semantics of the column are not matchable, not when matching is not feasible due to the representation. E.g., there are many different systems for classification of objects so that a given tables classes may not be correlatable with others in practice. However such a column should not be given a property of local, since the base concept of what a class is, is shared by other tables and one could imagine usefully joining tables on this idea. However, the jitter in the temperature of the fourth stage photomultiplier of some instrument is just not something that can usefully be compared with data in a VO context and should be given the attribute local.

This fills a potentially serious gap in the original proposal. The idea was (I think) that for say a bunch of engineering data a user would just specify a UCD of (say) instr. However we now need to have a rule for understanding how long a UCD needs to be before it makes sense to compare them. In this proposal the UCD is given as instr.local so we still give a bit of information about what is in the table, but we make it clear that the column shouldn't even be thought of for cross-matching.

4.3 Freedom in Hierarchy

UCDs at any level can be used to describe some parameter, whether sub-levels are existing or not. This rule implies that we do not use a qualifier like `misc` or `gen`: if a quantity is not accurately defined, we just use the `parent' UCD. An example comes with the division of the electromagnetic spectrum: the standard UCD words can label parts of the spectrum, for example `em.IR.3-4um` and `em.IR.4-8um`. To label a region from 3 to 5 um, the recommended UCD is the generic `em.IR`.

4.4 Standard Usage

The elements of the tree make use of a standard vocabulary, in the sense that a single word is used to designate a physical concept or quantity. For instance, if `electron` is used to designate any electron-related quantity, we write e.g. `phys.temp.electron` to designate the electronic temperature and not an abbreviation like `phys.temp.el`; conversely, `electron` keeps the same meaning among all UCDs. We should try to maintain a list of these meanings -- we are for instance using `temp` for temperature, `phys` for physical, and so on.

4.5 UCDs and Relationships among Concepts

UCDs describe relationships among concepts in three distinct ways.

UCDs provide a standard framework in which we can create labels whose relationship is understood as expert information in the field. [Doubtless the word ontology belongs in this paragraph.] E.g., there is nothing in the two strings pos.eq.ra and pos.eq.dec that indicates the relationship between Right Ascension and declination. However users and software understand that relationship and UCDs provide consistent markers to enable the use of that knowledge. This is perhaps just a fancy way of saying that UCDs provide a consistent way of labeling columns – unlike column names which tend to vary wildly.

UCDs also indicate the comparability of columns. Two columns with the same UCD should be 'comparable' in some sense. The more precise the UCD is, the more useful this comparability is likely to be.

The next section uses UCDs of objects, usually with the attribute instance. Comparability of such UCDs does not generally mean that they have the same structure, but rather that there is some sense that the underlying instances may be the same. E.g., two source tables may both have a UCD of src.instance but one may contain flux information while the other has proper motions. These tables are comparable, because rows of the two tables might be referring to the same object. I.e., the underlying source instance is the same even though the attributes of the source expressed in the tables are very different.

Columns may be comparable even if they have different UCDs. Generally if the two UCDs can be made identical by omitting terminal atoms in one or more words of the UCDs, then the fields are comparable when thought of in the more generic context of the truncated UCD. E.g., the UCDs phot.flux;em.optical.band.u;meas.error.statistical and phot.flux;em.xray.hard;meas.error.systematic may be truncated to phot.flux;em;meas.error and compared as instances of this more generic UCD.

Similarly if UCDs differ in their modifiers they are comparable as UCDs where the differing contexts have been omitted. E.g., a calculated and measured flux might have UCDs phot.flux;em.optical;intent.calculated;value and phot.flux;em.optical;value. These fluxes are comparable by ignoring the intent of the first column.

When modifiers or atoms in the UCD words are ignored software and users should be aware that entity is being treated in a more generic context than it was defined in.

The final way that UCDs describe relationships is in the association of different attributes to a given base concept (or base concept and modifiers). The attribute structures define generic relationships among entities, e.g., the relationship between a value and an error or between an image and a smoothed counterpart.

To build an understanding of data, we need to address not only the semantic relationships among concepts that UCDs provide: i.e., the flux concept is related to errors in fluxes, but also a physical relationship: this particular flux is related to this particular error. How this is done, and how the combination of semantic and physical relations allows use to build models of VO datasets is described in the following section.

Just a few words here to summarize the distinction between this proposal and the proposal it responds to…

The basic difference is that everything revolves around the base concepts, not properties. Here are some specific differences:

The function of a word is always the same. It cannot sometimes be a property and sometimes a concept.

Many UCDs that would valid in the old scheme are illegal in this one. Indeed I'm not sure that any string of words can be determined to be illegal in the old scheme. The only restriction on words was that the first be a property, and there is no way to tell which words are properties. This strictness is a good thing if we anticipate using UCDs in software. E.g., stat.error was a valid UCD even though it was completely unclear what it referred to. UCDs could be built into arbitrarily large agglomerations. This is much harder in the new scheme, since one cannot put in a second base concept, modifiers are strictly ordered and most of the attributes tend to be limiting. E.g., the statistics trees decimates the dimensionality of the quantity so it's hard to imagine UCDs like concept;stat.max;stat.max even though they may be lexically valid. In the previous scheme UCDs like:

arith.diff;arith.sum;phot.flux;em.optical;arith.sum;phot.flux;em.optical;phot.flux.xray

were clearly in the offing with hosts of parenthesis and such. All of this complex tying of one table to another has been moved to where it belongs in the grouping structures discussed in the next chapter.

The format for UCDs is much more tightly constrained and it should be much easier to ensure that UCDs for the same quantity are actually written identically. The previous scheme allowed substantial ambiguity in how a UCD should be written even if the words comprising the UCD were known. This will make it easier to build UCDs and to automatically parse them.

Only a single base concept can appear in a UCD. Relationships among base concepts are addressed using the grouping structure discussed in the following chapter.

The order of properties and concepts is reversed and mirrors the conventional representations of objects and attributes.

Error is moved to a new meas tree. I would suggest this even for the previous scheme. Errors are not a statistical property – they arise from the physics associated with the measurement. This also allows a useful separation of stat and filter attributes.

The local property is added. Something similar would be appropriate in the old scheme.

Several new word trees are defined as well as a few special single word attributes. The latter are important for the discussion in section 5. Most of the trees are simple reflections of what I see as holes in the UCD coverage but they probably aren't critical and have nothing really to do with the difference in schemas. I haven't actually seen the full tree of UCDs for the new proposal. One important issue with my proposal is that I would like users to be able to distinguish attributes, values and concepts by looking only at the first atom of a given word and not repeat words in those three categories. This is manifestly possible (e.g., I could simply create new trees as needed), and I think it even works out nicely, but the proof is in the pudding.

The alias concept is deleted. The whole idea of UCDs is to be standard and so I think it's a bad idea. but perhaps I misunderstand it.

5. UCDs in Complex Structures.

This following discussion is largely independent of whether we choose the original UCD2 framework or what I have described above. However this approach enables us to build complex concepts from the simple UCDs so that the UCDs themselves bear much less weight in describing the organization of data. Personally I believe this approach will largely obviate the need for utypes in VOTables. I would be very interested in other reactions.

The recent proposal for grouping structures within VOTables makes it possible to use UCDs in a much more sophisticated way than was possible when tables could comprise only a linear array of columns. UCDs are not necessarily tied to VOTables, but we recommend that grouping structures should be available and used whenever UCDs are used in other contexts to describe tabular information. From the perspective of UCDs the key elements of the group are that it 'physically' associates a group of columns, and that there is a group UCD. A UCD for tables as a whole will also be extremely valuable.

When we say that the group physically associates entities, all we mean is that by placing entities within a group the table builder is declaring that these entities are linked together in some relationship. The grouping does not describes the semantics of the relationship. That is the role of UCDs. Generally the grouped entities may be thought of as attributes of the entity described in the grouping UCD.

Groups should be used in a table whenever two or more columns (or parameters) are related to one another in some way other than as being distinct attributes within their enclosing table (or group). Table and group UCDs are usually simple two word UCDs. The use of groups and table UCDs is described below using a series of examples. These examples use a simple graphic, the *UCDtree* to represent the UCDs within a table. Each UCD is placed on its own line occasionally with a comment following. Whitespace separates the UCD from any comment. The table UCD is on the first line and is not indented. Whenever we go into a group or table the subsequent lines are indented one more tab. This immediate illustrates the group structure of the table.

.

Example 1: A simple table giving the id, position and flux of a set of sources.

src;instance	The table UCD, since this defines a group subsequent lines are indented
meta.id;value	A column UCD
pos;instance	A group UCD – since the next column is indented more
pos.eq.ra;value	A column UCD
pos.eq.dec;value	A column UCD
phot.flux;value	A column UCD

This is the UCDTree for a simple table with an id, position and flux for a set of sources. The table UCD indicates that each row in the table represents one source. The ID and flux are given using simple value UCDs, but since RA and Dec are related to each other they are grouped as a position instance.

The VOTable XML for the header of this table might look something like:

```
<TABLE … ucd='src;instance'>
   <FIELD … ucd='meta.id'/>
   <GROUP … ucd='pos;instance'>
      <FIELD … ucd='pos.eq.ra'/>
      <FIELD … ucd='pos.eq.dec'/>
  </GROUP>
  <FIELD … ucd='phot.flux'/>
…
</TABLE>
```

Only the UCD relevant elements of the XML table have been given. Note the use of the convention allowing the omission of the value attribute.

Software that knows nothing of group UCDs will work fine here. It will find the RA and Dec and flux columns without any trouble. We don't lose anything by using groups.

Now let us consider more complex cases, to see the real power of combining UCDs and groups. In the old UCD schema UCDs with 'main' attributes are needed to resolve ambiguities among columns with similar UCD, but this kludge fails as tables get even modestly complex.

Example 2: A source table with the observation plate center and an observation table with a guide star position.

src.instance	This is the source table
meta.id;value	The id of the source
pos.instance	
pos.eq.ra	The RA for the source
pos.eq.dec	The Dec for the source
obs.instance	
meta.id;value	The ID of the plate
pos.instance	
pos.eq.ra	The RA for the plate center
pos.eq.dec	The declination for the plate center
phot.flux;value	The flux of the source

obs.instance	This is the observation table
src.instance	
meta.id;value	The id/name of the guide star
pos.instance	
pos.eq.ra;value	The RA of the guide star
pos.eq.dec;value	The declination of the guide star
meta.id;value	The id of the observation
pos.instance	
pos.eq.ra	The RA of the center of the observation
pos.eq.dec	The declination of the observation.
time.exposure;value	The exposure time of the observation

These two UCDTrees merit careful study. This example is contrived such that, with the exception of the last column, the column UCDs in these two tables are identical and in the same order yet both software and humans should have no trouble distinguishing the very different semantics of the two tables.

First, the table UCDs indicate the basic sense of what each row in the tables is. Software or people can see that we get information about sources in the first table, and information about observations in the second.

Next, to find the primary position (or id) fields we simply look for the position field which is least indented, closer to the root table UCD. This simple approach will obviate the need for 'main' qualifiers. Note that the order of the columns is not significant – it is the table structure that reveals the primary fields.

Finally consider a situation where a user is looking for the positions of sources and does not care if the information is a primary attribute of the table. There's no problem with the first table naturally – it is a source table after all. But the second table manifestly has such information too and the user can easily find it. What is quite remarkable is that user can get the source positions in a non-source table by making exactly the same query of the second table as of the first: find a pos entry in a src context. Essentially the user can specify a model – a data model – for the information they are looking for and find it regardless of its depth in the table. This is exactly the kind of query on hierarchical data that XQuery has been developed for.

Example 3: Two examples of two sources in a table.

src.instance
    phys.wavelength;value
    phot.flux;value
    src.instance
        phot.flux;value

---

src.multiplet
    phys.wavelength;value
    src.instance
        phot.flux;value
    src.instance
        phot.flux;value

Here we have two tables with three columns with identical UCDs, but the grouping UCDs are quite different. What could this mean?

The tables seem to be spectra since we have a flux and wavelength for each row. In the first case there is also a source which is an attribute of the table source.

In the second there are two sources that are not dependent on one another. The multiplet attribute of the table UCD suggested that we would have more than one instance in each row and the detailed examination of the UCDTree confirms this. Apparently the wavelength column is shared by each of these two source instances.

The first table would be appropriate where the second flux value is from a calibrator object. The calibration object is considered as an attribute of the primary source. So if we want to find the flux for the primary source we look at the less indented column.

The second table would be appropriate if we happen to have a detector that takes two spectra of independent targets and records them together. The two sources are associated physically in the table – they were observed at the same time perhaps -- but there is no semantic relationship between them, or at least none that the table creator chose to express. It is important that our semantic framework is able to indicate lack of relationship just as easily as relationship.

One interesting issue here is what we choose for the table UCD. Here the UCD indicates that this is information about a source, but it could perhaps have used a phot.flux;instance or phot.flux;multiplet. A case can be made for both but we should try to agree on appropriate templates to be used within the VO.

Example 4: Source cross-correlations.

One of the commonest operations we anticipate in the Virtual Observatory is the correlation of two object tables. We can make quite subtle distinctions about the results of such a correlation by how we structure the resulting output. Consider

src.multiplet		
	src.instance	
		meta.id;value The Sloan ID
		pos;instance The Sloan position (the actual columns are suppressed)
		phot.flux;em.optical;value A Sloan Flux
	src.instance	
		meta.id;value The 2MASS ID
		pos.instance The 2MASS position
		phot.flux;em.infrared;value The 2MASS flux
	phys.degrees;value	The offset between the two targets

This representation of the table suggests that each row is a candidate pair that we have associated, but by making the table UCD a multiplet we have *not* implied that the two sources in each row necessarily refer to the same object. Note that we cannot find a 'main' position for this table. This is good. In this situation we don't really know what the correct position is. Structure should not manufacture information that does not exist and it is important that the structure allows us to preserve that ambiguity.

If we believe that the cross-match is matching data from the same object we might have written data in a table with a UCDTree

src.instance		
	meta.id;value	The Sloan ID
	pos;instance	The Sloan position (the actual columns are suppressed)
	phot.flux;em.optical;value	A Sloan Flux
	src.instance	
		meta.id;value The 2MASS ID
		pos.instance The 2MASS position
		phys.degrees;value The offset between the two targets
	phot.flux;em.infrared;value	The 2MASS flux

Aside from moving the offset column we haven't changed the data in the table at all. However the table now indicates that each row contains data about a single source. The Sloan ID and position are now the 'main' name and id. The 2MASS information is retained, but other than the flux value it is considered subsidiary to the Sloan. The software that has done the cross-correlation has resolved the ambiguities that arise in the cross-match.

Example 5. Cross-correlating sources and observations.

A generic cross-correlator might be asked to correlate a table of observations and a table of sources. Perhaps the user wants to find the sources within the field of view of the observations, or perhaps they wish to find the observations that contain a source. Without knowledge of what the user wants the correlator might produce a UCDTree like

concept;multiplet
    obs.instance
        … observation fields
    src.instance
        … src fields

Note the use of concept in the table UCD. Given more input about the user wants, the correlator might have chosen either src.instance or obs.instance as the basic type of the table.

In the discussion of the last few examples above we began abbreviating UCDTrees by not including every column but just including the group parameters in some places. This illustrates another aspect of the flexibility of the UCDTree in describing the table structure. We needn't clutter the discussion of our data models with irrelevant details, e.g., we can talk about positions with having to specify the detailed columns. We can just as easily match a position specified in galactic coordinates as one specified in RA and Dec.

5.1. UCDTrees and Data Models

Standard idioms for the structure of tables should be developed as soon as the new UCD scheme and VOTable grouping enhancements are available. If tables use the grouping structures in consistent fashions, then UCDTrees may be used in defining data models at least as they relate to tabular data. Since most common high-level datatypes can be represented in a number of different ways, data models will likely need to specify multiple possible structures, but the with the development of tools specifically intended for winkling commonality hidden within complex XML structures, notably XQuery, it may be possible to build tools that see if tables can be used as spectra, timeseries, images and so forth. Even if this ambitious goal cannot be met, the structuring of tables allows data models to unambiguously find required information, e.g., the error associated with position or flux or the reference frame parameters associated with a given measurement when there are several reference frames used within the table.

Unlike UCDs where it seems reasonable to strive for a syntax that largely ensures the uniqueness of the appropriate UCDs, it seems unlikely that there is any natural way to ensure unique table structures for complex data. The publication of standard UCDTrees or other structure templates will be needed to ensure that VO services can translate data models into actions on real data.

I have deleted the extensive description of the previous scheme. While there was a lot of text that is probably clearer and more precise than what I have written, I wanted to get this new proposal out as quickly as possible, and trying to express what I wanted to say in the context of the existing document was harder than just doing things directly. However I would anticipate that in a final document substantial elements might be retained. This material was very important in clarifying my thoughts on UCDs even though I came to somewhat different conclusions.

The discussion on use cases and self-match has been deleted. Self-match issues have been discussed above and the old text while defining the issues says nothing about how they are to be resolved.

I have deleted the section on software and services. It was quite sketchy anyway and I'm not sure it adds anything to the document.

6. UCD Steering Committee

6.1 Creation of a Board for New UCD Words

We believe that the inclusion of new UCD words must be a flexible process, yet controlled. The best way to accomplish these two needs would be to create a proper scientific board that would study new UCD requirements and, within a given period of time, give an answer as to whether a new UCD must or must not be included in the UCD standards.

The use of "mission-specific" namespaces has been addressed in many occasions, and we believe that namespaces should be avoided as much as possible. There has been an exercise in revising the VOX words for the SIAP protocol and trying to assign existing UCDs to them, or proposing new UCD words for the non-existing ones.

The responsibility of the board would consist of studying the cases where a UCD word is proposed and to figure out whether the proposed word should be accepted or rejected, and in case of rejection recommending the closest existing word that should be used.

In case a new word is accepted into the main tree, an internal procedure should be established so that the new UCD becomes live after a proper internal new release in a short period of time.

It should be agreed whether this board would study the proposed cases in an "on demand" basis or would collect requests and study them on a periodic basis.

A suggestion on the formation of this scientific committee would be that it might contain people from CDS (as they have the experience and the resources) but it should be offered to all relevant parties. It would also be very important to have a member from the data providers community, as the scientists view on some issues might not include other important views from data providers.

6.2 A procedure to request new UCD words

A procedural document should be created to make it easy to a user to ask for a new UCD and to understand the implications of doing so. This document would address:

- the contact point to ask for new UCD
- the life-cycle of the process of asking for a new UCD

- when and how a new UCD becomes live
- what to do if a UCD is rejected

This type of actions could (and should) be supported by tools like an automatic form that is filled in and sent to the scientific board, giving an answer back to the user acknowledging the request, and giving a time estimate for an answer. All these issues will be suggested in a separate point.

Lessons should be learnt from other projects where similar boards exist. There should be a thorough investigation (maybe from the board mentioned above) of how other projects have worked in this direction (like the Planetary Data System (PDS), the FITS consortium, the W3C) and try to get the right things from them while avoiding the wrong ones.

6.3 Creation of a Technical Board

There should be tools available for the user to check for the existence of UCDs, etc. Some of these tools exist already in CDS, and they are good candidates to become the sort of "official" tools for the UCD standards. However, we feel it is necessary to have a proper technical board that could, eventually, decide on what tools are really necessary to make the UCD work feasible and as easy as possible for the user. This board would be mainly in charge of writing proper requirements for the tools. The management of resources, etc., should be handled by the concepts wanting to work for the VO project, but the definitions of requirements, etc., should be centralized on this board.

6.3 Contact point for UCD issues

We feel the necessity to create a contact point to which all UCD related matters can be addressed. This contact point could be a web address devoted explicitly to that in the context of the VO, a properly organized web place, where all the tools would be available, as well as all documents and procedures for creation of new UCD words, etc., with practical examples and the like.

Appendix 1.

There has been much debate in the UCD forum over division of the electromagnetic spectrum, since this is where the qualitative and the quantitative meet. If we put every word about spectrum coverage into the UCD, there would be hundreds of terms, therefore we have chosen to keep to a rational division (below) plus a very few special words.

The wavelength spectrum is first divided in the 7 classical domains radio / IR / Optical / UV / EUV / X-ray / gamma. Further divisions are made to define the large bands classically used in optical / IR / UV, and in radio frequencies we keep bands spaced by a factor 2. In Figure 3, a special word is there for `Halpha` as a subclass of `opt.R`. If a desired band does not fit in the rational list, it is recommended to use the smallest enclosing band.

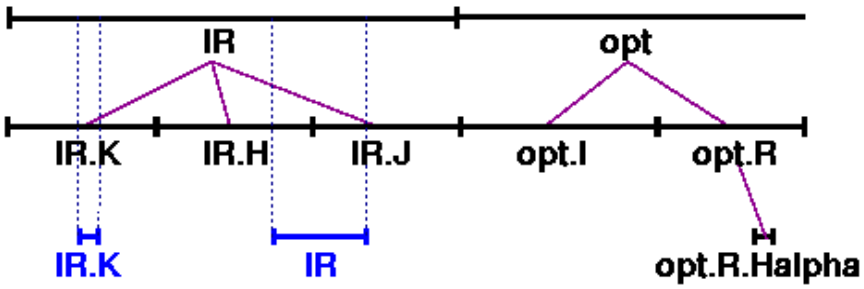

Figure 1: Hierarchical organization of the electromagnetic spectrum. The standard bands are represented in black. The suggested description of the non-standard blue ranges is shown in blue: in each case, we use the smallest enclosing standard band.

The overall list is as follows:

UCD designation	Lambda	Freq	Energy	Notes
Radio Regime				
em.radio.20-100MHz	>3m	<100MHz		
em.radio.100-200MHz	1.5-3m	100-200MHz		
em.radio.200-400MHz	75-150cm	200-400MHz		
em.radio.400-750MHz	40-75cm	400-750MHz		
em.radio.750-1500MHz	20-40cm	750-1500MHz		
em.radio.1500-3000MHz	10-20cm	1.5-3GHz		
em.radio.3-6GHz	5-10cm	3-6GHz		
em.radio.6-12GHz	2.5-5cm	6-12GHz		
em.radio.12-25GHz	1.2-2.5cm	12-25GHz		
em.radio.25-50GHz	6-12mm	25-50GHz		
em.radio.50-100GHz	3-6mm	50-100GHz		
em.radio.100-200GHz	1.5-3mm	100-200GHz		
em.radio.200-400GHz	750-1500µm	200-400GHz		
em.radio.400-750GHz	400-750µm	400-750GHz		
em.radio.750-1500GHz	200-400µm	750-1500GHz		COBE 240µm
em.radio.1500-3000GHz	100-200µm	1500-3000GHz		COBE 140µm
Infra-Red Regime				
em.IR.60-100um	60-100µm	3-5THz		IRAS 100µm
em.IR.30-60um	30-60µm	5-10THz		IRAS 60µm
em.IR.15-30um	15-30µm	10-20THz		IRAS 25µm
em.IR.8-15um	8-15µm	20-37.5THz		N band; IRAS 12µm
em.IR.4-8um	4-8µm	37.5-75THz		M band;
em.IR.3-4um	3-4µm	100-150THz		L, L', L"
em.IR.K	2-3µm	75-100THz		K band
em.IR.H	1.5-2.0µm	200-300THz		H band;

em.IR.J	1.0-1.5um	150-200THz		J band;
Optical Regime				
em.opt.I	750-1000nm	300-400THz	1.2-1.6eV	I band;
em.opt.R	600-750nm	400-500THz	1.6-2.0eV	R band;
em.opt.V	500-600nm	500-600THz	2.0-2.4eV	V band;
em.opt.B	400-500nm	600-750THz	2.4-3.0eV	B band;
em.opt.U	300-400nm	750-1000THz	3.0-4.0eV	U band;
Ultra-Violet Regime				
em.UV.200-300nm	200-300nm	1000-1500THz	4-6eV	UV1 band
em.UV.100-200nm	100-200nm	1500-3000THz	6-12eV	UV2 band;
Extreme Ultra-Violet Regime				
em.EUV.50-100nm	50-100nm	3-6PHz	12-24eV	Lv{Limit}=91.2nm
em.EUV.10-50nm	10-50nm	6-30PHz	24-120eV	
X-ray Regime				
em.X-ray.soft	6-100Å	30-500PHz	0.12-2keV	
em.X-ray.hard	0.1-6Å	0.5-30EHz	2-12keV	
Gamma Regime				
em.gamma.soft	0.25-10pm	30-1200EHz	12-500keV	
em.gamma.hard	<250fm	> 1200EHz	>500keV	e+/e-