Appendix 1
A Summary of IVOA Standards

Alan Grainger
University of Leeds

1. Introduction

The ASTROTROP project's evaluation of the feasibility of using AstroGrid software in a Pan-Tropical Virtual Observatory concluded that this software cannot be used directly, or easily adapted for such use, because astronomical standards are built into the software code. Consequently, the evaluation recommended that the TROPGLOBE network of tropical forest researchers should *imitate* the AstroGrid approach by: (a) using existing generic geospatial software, and (b) agreeing on their own standards to make this software functional for their needs.

It is proposed that priority be given to agreeing on a limited set of standards in the first instance. This set will be roughly based on the priority set of standards agreed by the International Virtual Observatory Alliance (IVOA), but modified to meet specific TROPGLOBE requirements. Six high priority standards were chosen from a total of 48 existing IVOA standards. They are listed below, in the likely order of discussion at the Conference, and are grouped as in the corresponding IVOA Working Groups, in case the latter prove convenient for TROPGLOBE Working Groups too:

a. Semantics: *Vocabularies; UCD Controlled Vocabulary*.
b. Registry: *Resource Metadata; Registry Interfaces*.
c. Applications: *VOTable Format*
d. Data Access Layer: *Simple Search*
e. Data Model: -
f Grid & Web Services: -

This document, which has been prepared for discussion and revision at the 2nd ASTROTROP Conference, provides an accessible summary of the corresponding IVOA standards, to help TROPGLOBE members to frame their standards. Since IVOA standards are scattered across multiple documents, it is difficult for people new to IVOA to get an overview of what is involved. This document aims to provide such an overview by combining key standards documents in a single piece of text.

Because most IVOA standards documents are clearly written, environmental scientists will find it easy to understand the original text, so this has been directly quoted - in italics, for convenience. Most astronomical examples and terms have been deleted where this does not obstruct the meaning, but some have been retained to provide "markers" for substituting environmental terms. To provide context, in each section the Introduction is followed by a sub-section which specifies what tropical forest researchers need to do next to adopt this approach to their needs.

2. Vocabularies (Semantics)

2.1 Introduction

The first group of standards includes basic terms on tropical forests. Defining terms is the first action in any academic enterprise, but is particularly important here because the tropical forest research community, unlike the astronomical community, consists of multiple disciplines, each of which may use different terms for the same meaning, and the same term for different meanings. After highlighting some key tropical forest terms, this section quotes - in italics - relevant parts of "Vocabularies in the Virtual Observatory (Vs 1.19) which is found at http://www.ivoa.net/Documents/REC/Semantics/Vocabularies -20091007.html. As with the IVOA, using existing forest vocabularies is preferable to attempting to produce new definitions of terms.

2.2 Actions for Tropical Forest Researchers

Tropical forest researchers need to identify existing vocabularies that will be useful for their needs and to provide clear definitions of such important terms as:

a. Forest, tree, shrub, bush.
b. Forest area, tree cover, tree density.

2.3 Overview

"This document specifies a standard format for vocabularies based on the W3C's Resource Description Framework (RDF) and Simple Knowledge Organization System (SKOS). By adopting a standard and simple format, the IVOA will permit different groups to create and maintain their own specialised vocabularies while letting the rest of the community access, use, and combine them. The use of current, open standards ensures that VO applications will be able to tap into resources of the growing semantic web.

Information of relevance to the Virtual Observatory (VO) is not confined to quantities easily expressed in a catalogue or a table. Fairly simple quantities are easily manipulated and stored in VOTables and can currently be identified using IVOA Unified Content Descriptors (UCDs). However, concepts and quantities use a wide variety of names, identifications, classifications and associations, most of which cannot be described or labelled via UCDs.

There are several basic forms of organised semantic knowledge of potential use to the VO. A slightly formal structure is a "vocabulary", where the label is drawn from a predefined set of definitions which can include relationships to other labels; vocabularies are primarily associated with searching and browsing tasks. At the other extreme are "ontologies", where the domain is formally captured in a set of logical classes, typically related in a subclass hierarchy.

An ontology is necessary if we are to have a computer (appear to) "understand" something of the domain. There are distinct use cases for letting human users find resources of interest through search and navigation of the information space. The most appropriate technology to meet these use cases derives from the Information

Science community, namely that of controlled vocabularies, taxonomies and thesauri. In the present document, we do not distinguish between controlled vocabularies, taxonomies and thesauri, and use the term vocabulary to represent all three. A number of vocabularies have been created in our field, with a variety of goals and intended uses.

2.4 Use-cases, and the motivation for formalised vocabularies

This standard establishes a set of conventions for the creation, publication, use, and manipulation of vocabularies within the Virtual Observatory, based upon the W3C's SKOS standard. The goal of this standard is to show how vocabularies can be easily expressed in an interoperable and computer-manipulable format, and the first normative section of this Recommendation contains requirements and suggestions intended to promote this.

It is not a goal of this standard to produce new vocabularies, or substantially alter existing ones; instead, SKOS versions of existing vocabularies (normative) are directly and mechanically derived from existing vocabularies. It therefore follows that the ambiguities, redundancies and incompleteness of the source vocabularies are faithfully represented in the distributed SKOS vocabularies. We hope that this formalisation process will create greater visibility and broader use for the various vocabularies, and that this will guide the maintenance efforts of the curating groups.

2.5 Formalising and managing multiple vocabularies

Where there are multiple vocabularies in use, one approach is to create a single consensus vocabulary, which draws terms from the various existing vocabularies to create a new vocabulary which is able to express anything its users might desire. The problem with this is that such an effort would be very expensive, both in terms of time and effort on the part of those creating it, and to the potential users, who have to learn to navigate around it, recognise the new terms, and who have to be supported in using the new terms correctly (or, more often, incorrectly). The alternative approach to the problem is to evade it, and this is the approach taken in this document. Rather than deprecating the existence of multiple overlapping vocabularies, we embrace it, help interest groups formalise as many of them as are appropriate, and standardise the process of formally declaring the relationships between them. This means that:

a. The various vocabularies are allowed to evolve separately, on their own timescales, managed either by the IVOA, individual working groups within the IVOA, or by third parties.

b. Specialised vocabularies can be developed and maintained by the community with the most knowledge about a specific topic, ensuring that the vocabulary will have the most appropriate breadth, depth, and precision.

c. Users can choose the vocabulary or combination of vocabularies most appropriate to their situation, either when annotating resources, or when querying them.

d. We can retain the previous investments made in vocabularies by users and resource owners.

2.6 SKOS-based vocabularies (informative)

In this section, we introduce the concepts of SKOS-based vocabularies, and the technology of mapping between them.

2.6.1 Selection of the vocabulary format

After extensive online and face-to-face discussions, the authors have brokered a consensus within the IVOA community that formalised vocabularies should be published at least in SKOS (Simple Knowledge Organization System) format, a W3C draft standard application of RDF to the field of knowledge organisation. SKOS draws on long experience within the Library and Information Science communities, to address a well-defined set of problems to do with the indexing and retrieval of information and resources; as such, it is a close match to the problem this document is addressing.

The only technical distinction relevant to this document is that between vocabulary and thesaurus: BS-8723-1 defines a controlled vocabulary as a "prescribed list of terms or headings each one having an assigned meaning" [noting that "Controlled vocabularies are designed for use in classifying or indexing documents and for searching them."]; and a thesaurus as a "Controlled vocabulary in which concepts are represented by preferred terms, formally organised so that paradigmatic relationships between the concepts are made explicit, and the preferred terms are accompanied by lead-in entries for synonyms or quasisynonyms."

2.6.2 Content and format of a SKOS vocabulary

A published vocabulary in SKOS format consists of a set of "concepts".

A SKOS vocabulary is essentially a list of SKOS 'concepts', plus some metadata. Each SKOS concept has some or all of the following features, of which only the first two are required:

a. A single URI representing the concept, mainly for use by computers (that is, it is not required to be human-readable). This is a syntactic requirement.

b. A single preferred label in each supported language of the vocabulary, for use by humans. This is required.

c. Alternative labels which applications may encounter, whether simple synonyms or commonly-used aliases.

d. Hidden labels which capture terms which are sometimes used for the corresponding concept, but which are deprecated in some sense. Very common mis-spellings might arguably be included here, but there is no clear best practice.

e. A "notation" code (such as 5.1.1).

f. A definition for the concept, where one exists in the original vocabulary, to give a meaning for the term. This need not be extensive, but it should indicate to someone using the vocabulary what the concept is intended to refer to, and how precise it is expected to be.

g. A scope note to further clarify a definition, or the usage of the concept. While the definition explains the meaning of the term in some abstract sense, the scope note provides practical usage hints to a user of the vocabulary.

h. A concept may also be involved in any number of relationships with other concepts. The types of relationships are:

i. Narrower or more specific concepts.
ii. Broader or more general concepts.
iii. Related concepts.

i. Only the URI and preferred label are required; all of the remaining features are optional. Although including these features is desirable in a newly-developed vocabulary, if one is converting an existing vocabulary to SKOS form, it may not be feasible or desirable to add missing information.

j. In addition to the information about a single concept, a vocabulary can contain information to help users navigate its structure and contents:

i. The "top concepts" of the vocabulary, i.e. those that occur at the top of the vocabulary hierarchy defined by the broader/narrower relationships, can be explicitly stated to make it easier to navigate the vocabulary.
ii. Concepts that form a natural group can be defined as being members of a "collection".
iii. Versioning information can be added using change notes.
iv. Additional metadata about the vocabulary, for example indicating the publisher, must be documented using the Dublin Core metadata set.
v. The SKOS standard describes a number of "documentation properties"; these should be used to document provenance of and changes to vocabulary terms.
vi. A set of mappings between the concepts in different vocabularies.

2.6.3 Mapping relationships between vocabularies

We require a mechanism to relate the concepts in the different vocabularies. There are four types of relationship provided to capture the relationships between concepts in vocabularies, which are similar to those defined for relationships between concepts within a single vocabulary:

a. Equivalence between concepts, i.e. the concepts in the different vocabularies refer to the same real world entity. This is captured with the RDF statement AAkeys:#Cosmology skos:exactMatch avm:#Cosmology which states that the cosmology concept in the A&A Keywords is the same as the cosmology concept in the AVM.

b. Broader concept, i.e. there is not an equivalent concept but there is a more general one. This is captured with the RDF statement AAkeys:#Moon skos:broadMatch avm:PlanetSatellite which states that the AVM concept "Planet Satellite" is a more general term than the A&A Keywords concept "Moon".

c. Narrower concept, i.e. there is not an equivalent concept but there is a more specific one. This is captured with the RDF statement AAkeys:#IsmClouds skos:narrowMatch avm:#NebulaAppearanceDarkMolecularCloud which states that the AVM concept "Nebula Appearance Dark Molecular Cloud" is more specific than the A&A Keywords concept "ISM Clouds".

d. Related concept, i.e. there is some form of associative relationship. This is captured with the RDF statement AAkeys:#BlackHolePhysics skos:relatedMatch avm:#StarEvolutionaryStageBlackHole which states that the A&A Keywords concept "Black Hole Physics" has an association with the AVM concept "Star Evolutionary Stage Black Hole".

2.7 Publishing vocabularies (normative)

A vocabulary which conforms to this IVOA standard has the following features.

a. Dereferenceable namespace. The namespace of the vocabulary MUST be dereferenceable on the web. That is, typing the namespace URL into a web browser will produce human-readable documentation about the vocabulary. In addition, the namespace URL SHOULD return an RDF version of the vocabulary if it is retrieved with one of the RDF MIME types in the HTTP Accept header.

b. Long-term availability. The files defining a vocabulary, including those of superseded versions, SHOULD remain permanently available. There is no requirement that the namespace URL be at any particular location, although the IVOA web pages, or a journal publisher's web pages, would likely be suitable archival locations.

c. Distribution format. Vocabularies MUST be made available for distribution as SKOS RDF files in RDF/XML format. A human readable version in Turtle format SHOULD also be made available. As an alternative to Turtle, vocabularies may be made available in that subset of Notation3 which is compatible with Turtle; if Turtle or Notation3 is being served, it is prudent to support both text/rdf+n3 and text/turtle as MIME types in the Accept header of the HTTP request.

A publisher MAY make available RDF in other formats, or other supporting files. A publisher MUST make available at least some humanreadabledocumentation.

d. Clearly versioned vocabulary. The vocabulary namespace SHOULD NOT be versioned, but it SHOULD be easy to retrieve earlier versions of the RDF describing the vocabulary.

e. Concepts must have labels. Each concept MUST have both a URI naming it, and a human-readable skos:prefLabel (the first is a syntactic requirement of SKOS, and so is trivially satisfied; the second is an extension to SKOS).

f. No restrictions on source files. This Recommendation does not place any restrictions on the format of the files managed by the maintenance process, as long as the distributed files are as specified above."

3. Unified Content Descriptors for a Controlled Vocabulary (Semantics)

3.1 Introduction

Unified Content Descriptors (UCDs) are used in AstroGrid "for exchanging information using a controlled vocabulary. They are used in the VOTable standard to attach a standard description to table column names, for example. The data providers do not need to change the internal descriptions of their existing databases. Nor is it required that people building from scratch a new VO-compliant service use UCDs in the core of their system" (Derrierre et al., 2008).

All UCDs have been divided into just 12 categories, labelled by the first "atom" of the UCD "word":

a. arith (arithmetics)
b. em (electromagnetic spectrum)
c. instr (instrument)
d. meta (metadata)
e. obs (observation)
f. phot (photometry)
g. phys (physics)
h. pos (positional data)
i. spect (spectral data)
j. src (source)
k. stat (statistics)
l. time (time)

UCDs permit easy communication between scientists from different languages and disciplines. ASTROTROP UCDs could, for example, be divided into categories depending on disciplinary differences, e.g. area, biodiversity, carbon etc., supplemented by common categories, e.g. statistics.

3.2 Actions for Tropical Forest Researchers

Tropical forest researchers need to identify existing environmental UCDs and/or classifications that will be useful for their needs and then build new UCDs from there.

3.3 Building UCDs

This section outlines the basics of building UCDs. It is a direct quote from "The UCD1+ controlled vocabulary", Version 1.23, IVOA Recommendation 2007 April 2, found at http://www.ivoa.net/Documents/REC/UCD/UCDlist-20070402.html.

"A UCD is a string which contains textual tokens called 'words', separated by semicolons(;). A word is composed of 'atoms', separated by periods(.). The hierarchy is as follows:

atoms --> words --> composed words

UCD1+ are either single words, or a composition of several words.

UCDs are "controlled" (through a process that is also indicated in the reference document above). Control is exercised at the level of words (UCD1+) and at the level of the vocabulary (atoms) used to form words. A consistent list of atoms will be maintained, making sure that the same atom always means the same thing, even if used in combination with different other atoms.

Atoms are defined following these guidelines:

a. Abbreviations are kept to a minimum, and only if the result is not ambiguous. (ra, dec are acceptable, but t is ambiguous: time and temperature are used instead.)

b. Atoms are not hyphenated. The separation is marked by a capital letter to help readability (position angle = posAng) unless the composed word has a well known acronym (signal to noise ratio = snr) or short form (standard deviation = stdev). There are only two exceptions to this rule: (i) the X-ray band (em.X-ray) and (ii) thefrequency / wavelength intervals defining regions of the e.m. spectrum (e.g. em.radio.3-6GHz).

The list of UCD1+ words is maintained by the UCD Scientific Board. All words are preceded by a 'syntax' code that can help in the process of building composed UCD1+.

a. The code "P" means that the word can only be used as "primary" or first word.

b. "S" stands for secondary: the word cannot be used as the first word to describe a single quantity.

c. "Q" means that the word can be used as first or secondary word.

d. "E" means a photometric quantity, and can be followed by a word describing a part of the electromagnetic spectrum

e. "C" is a colour index, and can be followed by two successive word describing a part of the electromagnetic spectrum.

f. "V" stands for vector. Such a word can be followed by another describing the axis or reference frame in which the measurement is done.

The full list of words can be found at http://www.ivoa.net/Documents/REC/UCD/UCDlist-20070402.html. These examples show the basic UCD principles:

Q | instr | Instrument
S | instr.filter | Filter
S | instr.fov | Field of view
S | instr.pixel | Pixel (default size: angular)
S | instr.plate | Photographic plate
Q | instr.plate.emulsion | Plate emulsion
P | meta | Metadata
P | meta.abstract | Abstract (of paper, proposal,etc.)
P | meta.bib | Bibliographic reference
P | meta.bib.author | Author name
P | meta.bib.fig | Figure in a paper
P | meta.bib.journal | Journal name
P | meta.bib.page | Page number
P | meta.bib.volume | Volume number
S | obs.atmos | Atmosphere, atmospheric phenomena affecting an observation
Q | obs.atmos.refractAngle | Atmospheric refraction angle
S | obs.calib | Calibration observation
S | obs.field | Region covered by the observation
S | obs.image | Image
Q | obs.observer | Observer, discoverer
Q | obs.param | Various observation or reduction parameter
S | obs.sequence | Sequence of observations, exposures or events

3.4 The Latest UCD Standard (UCD1+)

The following text is a direct quote from the IVOA Standard for Unified Content Descriptors, Version 1.1, IVOA Recommendation 2005-08-12, found at http://www.ivoa.net/Documents/REC/UCD/UCD-20050812.html.

3.4.1 Scope of UCD

The Unified Content Descriptor (UCD) is a formal vocabulary for astronomical data that is controlled by the IVOA. The vocabulary is restricted in order to avoid proliferation of terms and synonyms, and controlled in order to avoid ambiguities as far as possible. It is intended to be flexible, so that it is understandable to both humans and computers. UCDs describe astronomical quantities, and they are built by combining words from the controlled vocabulary.

A UCD does not define the units nor the name of a quantity, but rather "what sort of quantity is this?"; for example phys.temperature represents a 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' it 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 intended to be understandable to both humans and computers.

The major goal of UCDs is to ensure interoperability between heterogeneous datasets. The use of a controlled vocabulary will hopefully allow an homogeneous,

non-ambiguous description of concepts that will be shared between people and computers in the IVO.

3.4.2 Syntax

A UCD is a string that contains textual tokens that we shall call words, which are separated by semicolons (;). A word may be composed of several atoms, which are separated by period (.) characters. The order of these atoms induces a hierarchy. Standard UCDs, which are validated by the IVOA, can start with the ivoa: namespace, but this namespace is optional. The use of namespaces, indicated by the presence of a colon in the word, is possible, but should be avoided as far as possible. They should be used only temporarily, for words that are not yet included into the vocabulary validated by the IVOA, and they should be replaced by the standard word as soon as it is validated.

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

a. There should be no space characters within a UCD. Space characters or tabs must be removed for a UCD to be well written.

b. The UCD syntax is case-insensitive — all uppercase characters should be converted to lowercase before parsing.

c. It is ensured that for any two words at the same level in the hierarchy, one can't be a starting substring of the other. This means that xxx.abc and xxx.abd can exist, but not xxx.abc and xxx.abcd (the latter one having the former as starting substring).

The building blocks for UCDs are the words (like phys.temperature), not the atoms (like temperature). People trying to assign a UCD to an astronomical quantity should first describe in natural language what the quantity is, and then search the list of valid words for the best matching words in the UCD vocabulary. In most cases, one single word will be sufficient, and the UCD will simply be this word. Creation of new words is the responsibility of the UCD Scientific Board, and should occur when the vocabulary is missing some useful knowledge description. Atoms are only considered at this level (creation of a new word): once created, words become the basic elements from which UCD are built.

The following examples have legal UCD syntax:

a. pos.eq.ra;meta.main
b. meta.id;src
c. phot.flux;em.radio;arith.ratio
d. PHot.Flux;EM.Radio;ivoa:arith.Ratio

Notice that the last two UCDs are identical because of the case insensitivity and because the default namespace is optional.

3.4.3 Construction

The idea of building UCDs by combining simple words makes the vocabulary less complex and more flexible. The words must not be too simple/ambiguous or too specific.

In order to avoid ambiguities, each word of the vocabulary will have an associated definition in plain text (and possibly related keywords). Words like source or type should only be used in the vocabulary with a very clear definition, and restrict only to one meaning in the case of homonyms (source can be a priori mean an object in the sky, a program code, a bibliographic reference, ...).

For this reason, and also in order to group similar words, words are composed of atoms. The first atoms in a word generally help specifying the context, and help understanding the word without reading its definition (e.g. pos.galactic.lat is the latitude in galactic coordinates while pos.ecliptic.lat is the latitude in ecliptic coordinates).

3.4.4 Combining Simple Words

While it is possible to build a UCD as a combination of several simple words, the primary word carries most of the meaning as to "what the quantity is". After the primary word, subsequent words are arranged by decreasing importance.

In the proposed scheme, UCDs are built by adding words from left to right, with each new word specifying/qualifying the combination to its left. The most important words when comparing two UCDs are the first ones. People or software that don't want to manage composed UCDs can use only the first word of the composed UCD (called the primary word). This word must give a first-order description of the quantity that is being described. It can be used as in UCD1, with the only change that the underscore ( ) is to be replaced by a period (.) in the parsing.

Examples of UCD1+ and how they are built:

a. The maximum temperature of an instrument. This is a temperature, so the primary word will be: phys.temperature. This temperature is that of an instrument, so we specify it next: phys.temperature;instr. And finally, we add a third word to indicate that this is the maximum value of a phys.temperature;instr, giving the final UCD: phys.temperature;instr;stat.max

b. The error on a magnitude measured in the V band. The quantity is an uncertainty, so the primary word will be stat.error. This uncertainty applies to a magnitude, so we write stat.error;phot.mag. Then, we can specify the photometric band with another word, giving stat.error;phot.mag;em.opt.V.

3.4.5 Use Case in Database Access

UCDs will be used in practice for exchanging information using a controlled vocabulary. They are used in the VOTable standard to attach a standard description to table column names, for example. The data providers do not need to change the

internal descriptions of their existing databases. Nor is it required that people building from scratch a new VO-compliant service use UCDs in the core of their system.

What is needed for interoperation with other systems is a "translation layer" that is able to associate UCDs to the parameters that are used internally, so that the output of the service contains a standard description that can be interpreted by other VO services.

A first VO service describes internally the right ascension and declination with names RA and DEC. For sending data to another service expecting right ascension and declination as an input, it uses a translation layer to attach UCDs to its parameters. The second service also has a translation layer that can interpret UCDs into its own parameters.

Mapping by the translation layer can be done using XML files. For the second service above, the quantities corresponding to UCDs pos.eq.ra and pos.eq.dec are to be found in the database table Obs-Table, which has column names alpha and delta:

```
<?xml version='1.0'?>
<!DOCTYPE ucdToDb SYSTEM 'ucdToDb.dtd'>
<ucdToDb>
<ucd name="pos.eq.RA" table="Obs-Table" col="alpha" />
<ucd name="pos.eq.DEC" table="Obs-Table" col="delta" />
<ucd name= ... />
</ucdToDb>
```

3.4.6 Use Case in a Registry

A registry can contain descriptions of catalogues, with the associated UCDs. The benefit of having access to the contents in terms of UCDs is that it is possible to explore the contents of a catalogue more extensively than with simple keywords.

For example, a catalogue dedicated to very accurate measurement of proper motions and parallaxes will certainly put keywords for these, but it might also contain a column that measures a radial velocity. With UCDs assigned, this column could be identified and the catalogue selected for someone searching for radial velocities, even if this is not the primary goal of the catalogue.

It is however not necessary to describe every element of a dataset by UCDs. Only the most relevant columns need to have UCDs attached to them. Parameters used for internal processing by a service do not need to have UCDs attached.

Consider the catalogue above described with UCDs in a registry. A query by UCD allows one to locate this catalogue and find that it contains radial velocities. Once the resource is located, one can then send a query to this resource, either on its specific parameters or again using UCDs."

4. Resource Metadata (Registry)

4.1 Introduction

Metadata are crucial to users finding the data they need. As Hanisch (2007) states: "An essential capability of the Virtual Observatory is a means for describing what data and computational facilities are available where, and once identified, how to use them. The data themselves have associated metadata (e.g., FITS keywords), and similarly we require metadata about data collections and data services so that VO users can easily find information of interest. Furthermore, such metadata are needed in order to manage distributed queries efficiently." This section describes the approach taken in AstroGrid, and is a direct quote, suitably adapted, from "Resource Metadata for the Virtual Observatory, Version 1.12, IVOA Recommendation 2007 March 2", found at http://www.ivoa.net/Documents/REC/ResMetadata/RM-20070302.html. Many astronomical metadata headings are generically applicable to earth observation and the environmental sciences, and so can be used as starting points for refining by more specific headings.

4.2 Actions for Tropical Forest Researchers

Tropical forest researchers can use a number of the generic IVOA metadata categories for their own purposes, that they must also identify categories specific to their needs.

4.3 Architecture

"We propose a hierarchical system for metadata management. At the top level we require a minimum amount of information, sufficient primarily to note the existence of a resource and to describe who is responsible for it. At lower levels, the metadata are more extensive and complex, allowing for the description of query syntax, access protocols, and usage policies.

A resource is a general term referring to a VO element that can be described in terms of who curates or maintains it and which can be given a name and a unique identifier. Just about anything can be a resource: it can be an abstract idea, such as sky coverage or an instrumental setup, or it can be fairly concrete, like an organisation or a data collec-tion. This definition is consistent with its use in the general Web community as "anything that has an identity" (Berners-Lee 1998, IETF RFC2396). We expand on this definition by saying that it is also describable.

An organisation is specific type of resource that brings people together to pursue participation in VO applications. Organisations can be hierarchical and range greatly in size and scope. At a high level, an organisation could be a university, observatory, or gov-ernment agency. At a finer level, it could be a specific scientific project, space mission, or individual researcher. A provider is an organisation that makes data and/or services available to users over the network.

A service is any VO resource that can be invoked by the user to perform some action on their behalf. Associated with any service is descriptive metadata about the service. Metadata generally include information the user needs to determine if a service is of interest and how the service may be invoked. Specific types of metadata are

described below. Note that the service itself need not be aware of the metadata that describe it.

A query service supports a query/response protocol. The user submits a query to the service that may define characteristics of interest, and the service returns a set of information to the user.

A registry is a query service for which the response is a structured description of resources. The resources described by a registry may be of any type. The registry may support a query that allows the user to indicate which resources might be of interest.

All resources are described by metadata. Resource metadata are generic, high-level, and independent of any specific service. Resource metadata include

a. Identity metadata, which gives the resource a name and an identifier.

b. Curation metadata, which describe who supports the resource and its availability (i.e., version, release date).

c. Content metadata, which describe what kind of information is available (types of data, sky coverage, spectral coverage, etc.). Content metadata can be either general, applying to all resources, or associated more specifically with data collections and the services that deliver data from them.

Resource metadata are typically not queryable parameters in the underlying services, but rather they encompass information that now is simply "known" to users, or must be discovered through other means.

Organisations, data collections, and services can be considered as classes of resources that may each require additional metadata to fully describe it, but which are not shared by other classes. For example, a service description would need to include its inputs, outputs, and how it can be accessed. Service metadata, therefore, can be thought of as an extension of the general resource metadata: where as the resource metadata, through its content metadata, describes what is available, the service metadata de-scribes how to access it.

Resource metadata will be collected through resource registration services, e.g., web forms that present a resource curator with the requisite fields and enumerated lists, and construct a resource descriptor in a standard format (such as VOTable).

The most general resource metadata is similar in concept to the Dublin Core metadata definitions (http://dublincore.org/documents/dces/), and where possible DC metadata elements have been used. VO metadata elements that correspond directly to DC counterparts are noted. The Dublin Core elements Language and Relation are not currently used in the VO metadata.

4.4 Resource metadata concepts

Below we describe the concepts we believe are needed in the resource metadata. These concepts may be instantiated in a variety of standard forms, e.g. XML, UCD tags, or FITS keywords. A limited number of keywords are considered essential for a basic understanding of the resource, and are thus denoted as required. All others are optional, or may be applied to certain classes of resources only.

4.4.1 Identity metadata

Title (string) [Dublin Core] [Required]
Definition: A name given to the resource.
Comment: Typically, a Title will be a name by which the resource is formally known. Title should be an unabbreviated form (e.g., Hubble Space Telescope) rather than an acronym unless the acronym is so well known as to be part of standard usage. Publishers are encouraged, but not required, to define unique Titles.

ShortName (string)
Definition: A short abbreviation for the name given to the resource.
Comment: The ShortName will be used where brief annotations for the resource name are desired, such as in GUIs that might refer to many resources in a compact display. ShortName strings are limited to a maximum of sixteen characters.

Identifier (URI) [Dublin Core] [Required]
Definition: An unambiguous reference to the resource within a given context. The syntax for Identifiers is described in IVOA Identifiers in the IVOA document collection (http://www.ivoa.net/Documents/).
Comment: The URI corresponding to the resource.

4.4.2 Curation metadata

Publisher (string) [Dublin Core] [Required]
Definition: An entity responsible for making the resource available
Comment: Examples of a Publisher include a person or an organisation. Users of the resource should include Publisher in subsequent credits and acknowledgments.

PublisherID (URI)
Definition: The identifier for the entity responsible for making the resource available. The syntax for Identifiers is described in IVOA Identifiers in the IVOA document collection (http://www.ivoa.net/Documents/).
Comment: This item is optional; an ID for the publisher may not yet be established (e.g., if the publisher has not yet been registered).

Creator (string) [Dublin Core]
Definition: An entity primarily responsible for making the content of the resource.
Comment: Examples of a Creator include a person or an organisation. Users of the resource should include Creator in subsequent credits and acknowledgments. Creator is intended to refer to the organisation or individuals responsible for the intellectual content of the resource, and not the organisation or individuals who may

have developed the service by which the content is made available. Creator.Logo (URL)
Definition: A URL pointing to a graphical logo, which may be used to help iden-tify the information resource.

Contributor (string) [Dublin Core]
Definition: An entity responsible for making contributions to the content of the resource.
Comment: Examples of a Contributor include a person or an organisation. Users of the resource should include Contributor in subsequent credits and acknowledgments. Like Creator, Contributor is intended to refer to the organisation or individuals responsible for the intellectual content of the resource, and not the organisation or individuals who may have developed the service by which the content is made available. Also see the Guide-lines under Creator.

Date (string) [Dublin Core] [Required]
Definition: A date associated with an event in the life cycle of the resource. Typically, Date will be associated with the creation or availability (i.e., most recent release or version) of the resource. ISO8601 is the preferred format (YYYY-MM-DD).
Comment: Dates may be approximate (e.g., year only, year and month). When the resource is an organisation, Date should refer to the approximate genesis of the organi-sation. When the resource is a service, Date should refer to the implementation date or the date the service came available.

Version (string)
Definition: A label associated with the creation or availability (i.e., most recent release or version) of the resource.

Contact (string, e-mail address)
Definition: The e-mail address for contacting the persons responsible for the resource.
Comment: Contact is split into two components for clarity.

Contact.Name (string)
Definition: The name of the contact.
Comment: A person's name, "John P. Jones", or a group, "Archive Support Team".

Contact.Address (string)
Definition: The mailing address of the contact.
Comment: All components of the mailing address are given in one string, e.g., "3700 San Martin Drive, Baltimore, MD 21218 USA"

Contact.Email (e-mail address)
Definition: The e-mail address of the contact.
Comment: For example, "John.P.Jones@navy.gov", or "archive@datacenter.org".

Contact.Telephone (string)
Definition: The telephone number of the contact.

Comment: Complete international dialing codes should be given, e.g., "+1-410-338-1234".

4.4.3 General content metadata

Subject (string, list) [Dublin Core] [Required]
Definition: A list of the topics, object types, or other descriptive keywords about the resource.
Comment: Subject is intended to provide additional information about the nature of the information provided by the resource. Is this a catalog of quasars? Of planetary nebulae? Is this a tool for computing ephemerides? Terms for Subject should be drawn from suitable vocabularies or thesauri.

Description (string, free text) [Dublin Core] [Required]
Definition: An account of the content of the resource.
Comment: Description may include but is not limited to: an abstract, table of contents, reference to a graphical representation of content or a free-text account of the content. Thorough text descriptions are particularly encouraged in order to make text-based searches against the registries maximally useful. Description should emphasize what the resource is about, as other matters such as who created it, when it was created, and where it is located are described elsewhere in the resource metadata.

Source (string) [Dublin Core]
Definition: A bibliographic reference from which the present resource is derived or extracted.
Comment: The present resource may be derived from the Source in whole or in part. Recommended best practice is to use the standard bibcode (see http://cdsweb.u-strasbg.fr/simbad/refcode.html), where available. If no bibcode is available, Source should use a string or number conforming to a formal identification or citation system.

ReferenceURL (URL) [Required]
Definition: A URL pointing to additional information about the resource. In general, this information should be human-readable.

Type (string, list) [Dublin Core] [Required]
Definition: The nature or genre of the content of the resource.
Comment: Type includes terms describing general categories, functions, genres, or aggregation levels for content. VO Types include:

Type	*Description*
Archive	*Collection of pointed observations*
Bibliography	*Collection of bibliographic references, abstracts, and publications*
Catalog	*Collection of derived data, primarily in tabular form*
Journal	*Collection of scholarly publications under common editorial policy*
Library	*Collection of published materials (journals, books, etc.)*
Simulation	*Theoretical simulation or model*

Survey	*Collection of observations covering substantial and contiguous areas of the sky surveys with easy-to-use images.*
Background	*Background information.*
BasicData	*Compilations of basic facts about objects.*
Historical	*Historical information about astronomical objects.*
Photographic	*Publication-quality photographs.*
Organisation	*An organisation that is a publisher or curator of other resources.*
Project	*A project that is a publisher or curator of other resources.*
Registry	*A query service for which the response is a structured description of resources.*
Other	*A resource not described by any of the above types.*

This list is extensible. Resources providing more than one type of content should list all relevant types.

ContentLevel (string, list)
Definition: A description of the content level, or intended audience.
Comment: VO resources will be available to professional astronomers, amateur astronomers, educators, and the general public. These different audiences need a way to find material appropriate for their needs.

ContentLevel	*Definition*
General	*Resource provides information appropriate for all users.*
Research	*Resource provides information appropriate for professional-level research and graduate school education*
Amateur	*Resource provides information of interest to lay people.*
Informal Education	*Resource provides information appropriate for education at museums, planetariums etc.*

Relationship (string)
Definition: A resource may be related to another resource in a way that is important to document, so that associated services or duplicate copies may easily be located.

mirror-of	*The resource is a mirror of another resource. Information gathered from the resources is indistinguishable.*
service-for	*The resource is a service associated with a data collection.*
derived-from	*The resource is a derivative of another resource, e.g., a subset selected for a particular scientific interest, or a reprocessed data collection.*
served-by	*The resource (e.g., a data collection) can be accessed via another service resource.*

RelationshipID (URI)
Definition: The identifier of an associated resource. The relationship is described in the Relationship metadata element.

4.4.4 Collection and service content metadata

Facility (string, list)
Definition: The observatory or facility where the data was obtained.
Comments: Some resources are likely to hold data from multiple observatories. If just a few, this could be a list; if very many, just say "many".

Instrument (string, list)
Definition: The instrument used to collect the data.
Comments: Can be a specific instrument name (Wide Field/Planetary Camera 2) or generic instrument type (CCD camera).

Coverage (string) [Dublin Core, with modifications]
Definition: The extent of scope of the content of the resource.
Comment: The Dublin Core notion of coverage is too generic to be of much use in the VO, where we need more specific information. We propose to subset this element as follows:

Coverage.Spatial (string)
Definition: The geographical coverage of the resource.

Coverage.RegionOfRegard (float, decimal degrees)
Definition: RegionOfRegard is a single numeric value representing the angle by which a positional query against this resource should be "blurred" in order to get an appropriate match.

Coverage.Spectral (string, list)
Definition: The spectral coverage of the resource.
Comment: Spectral coverage at the resource level will be in terms of general spectral regions (in astronomical terms, gamma-ray, x-ray, extreme UV, UV, optical, infrared, radio).

Coverage.Spectral.Bandpass (string, list)
Definition: A specific bandpass specification.
Comment: Some resources and services may choose to give spectral coverage in more specific terms than the general spectral regions.

Coverage.Temporal.StartTime (string)
Definition: The earliest temporal coverage of the resource.
Comment: Temporal coverage specifications will be given in ISO8601 format. An empty value field implies that there is no known earliest temporal coverage.

Coverage.Temporal.StopTime (string)
Definition: The latest temporal coverage of the resource.

Comment: Temporal coverage specifications will be given in ISO8601 format. An empty value field implies that there is no known latest temporal coverage, i.e., that information continues to be added to the resource.

Coverage.ObjectCount (int)
Definition: The total number of objects, catalog entries, telescope pointings, etc., in the resource.

Resolution (float)
Definition: The resolution of the resource contents.
Comment: Resolution is divided into the following sub-elements:

Resolution.Spatial (float)
Definition: The spatial (angular) resolution that is typical of the observations, in decimal degrees.

Resolution.Spectral (float)
Definition: The spectral resolution that is typical of the observations, given as the ratio $\lambda/\Delta\lambda$ (so that higher spectral resolution has a larger number).

Resolution.Temporal (float)
Definition: The temporal resolution that is typical of the observations, given in seconds.

UCD (string, list)
Definition: A list of the UCDs (Unified Content Descriptors, http://cdsweb.u-strasbg.fr/doc/UCD.htx) represented in the resource.
Comment: Some large or complex resources will have hundreds of associated UCDs and are unlikely to be specified in the resource metadata. Users of the resource meta-data should not assume that an empty specification implies that the resource has no associated UCDs.

Format (string, list) [Dublin Core]
Definition: The physical or digital manifestation of the information provided by the re-source.
Comments: Typical values would be "image/fits", "image/gif", "text/plain", "text/html", "text/xml" (for VOTable), etc. MIME types should be used where available to specify digital information formats in order to utilize existing standards.

Other format values will be used to describe the physical medium of the information: CDROM, Digital Planetarium, Online, Presentation, Print, Slides, Video. Format specifications may be combined, as in "Video, video/mpeg" (both hardcopy video cassettes and on-line MPEG files) or "CDROM, image/fits, image/gif" (FITS and GIF images are available on-line and on CDROM hardcopy).

Rights (string) [Dublin Core]
Definition: Information about rights held in and over the resource.
Comment: Dublin Core uses Rights to describe copyright and other intellectual property rights issues. In the VO context Rights would describe access privileges, using the following values: public, proprietary, mixed.

4.4.5 Data and metadata quality assessment

Users of virtual observatory resources need some way to assess the quality of the data and of the associated descriptive information in the registry. Data quality is both subjective and quantitative, and data collections may have no single data quality metric. While the completeness and consistency of the resource metadata itself may be a reasonable indicator of the quality of the associated resource, this is at best a qualitative measure. The following metadata elements are intended to capture the most basic measures of data quality, and may well require extensions as VO usage practices evolve and become more sophisticated.

DataQuality (char)
Definition: An overall assessment of the integrity, consistency, and level of documentation concerning uncertainty estimates and calibration procedures, of the data provided by the resource. We suggest 3 general grade levels, plus codes for unknown or un-documented cases:

A Data are fully calibrated, fully documented, and suitable for professional research.
B Data are calibrated and documented, but calibration quality is inconsistent. Users are advised to check data carefully and recalibrate.
C Data are uncalibrated.
U Data quality is unknown. If a resource does not provide a data quality assessment, class U should be assumed.

ResourceValidationLevel (int)
Definition: A numeric grade describing the quality of the resource description and inter-face, when applicable, to be used to indicate the confidence an end-user can put in the resource as part of a VO application or research study. The allowed values are:

0 The resource has a description that is stored in a registry. This level does not imply a compliant description.

1 In addition to meeting the level 0 definition, the resource description conforms syntactically to this standard and to the encoding scheme used.

2 In addition to meeting the level 1 definition, the resource description refers to an existing resource that has been demonstrated to be functionally compliant. When the resource is a service, it is considered to exist and to be functionally compliant if use of the Service.InterfaceURL or Service.BaseURL responds without error when used as intended by the resource. If the service is a standard one, it must also demonstrate the response is syntactically compliant with the service standard in order to be considered functionally compliant. If the resource is not a service, then the ReferenceURL must be shown to re-turn a document without error.

3 In addition to meeting the level 2 definition, the resource description has been inspected by a human and judged to comply semantically to this standard as well as

meeting any additional minimum quality criteria (e.g., providing values for important but non-required metadata) set by the human inspector (see comment below).

4 In addition to meeting the level 3 definition, the resource description meets additional quality criteria set by the human inspector and is therefore consid-ered an excellent description of the resource. Consequently, the resource is expected to be operate well as part of a VO application or research study.

If no value is provided, level 0 should be assumed.

Comment: Unlike other resource metadata, ResourceValidationLevel values will most often be set by entities other than the resource provider.

ResourceValidatedBy (URI)
Definition: The IVOA identifier for the registry or organisation that set the value of Re-sourceValidationLevel.

Uncertainty.Photometric (float)
Definition: The uncertainty of the photometric measurements provided by the resource.

Uncertainty.Spatial (float)
Definition: The uncertainty of the positional measurements, provided by the resource.

Uncertainty.Spectral (float)
Definition: The uncertainty of the wavelengths provided by the resource, given in meters.

Uncertainty.Temporal (float)
Definition: The uncertainty of the temporal measurements provided by the resource.

4.4.6 Service metadata concepts

The metadata necessary for describing a service will vary depending on the type of service it is. We propose two general categories of service metadata:

Interface metadata, which describe how to access the service—the inputs and the outputs. There will be standard types of interfaces that could include a web-browser-based interface (i.e., HTML Forms), a Web Service interface (describable by a WSDL docu-ment), a general HTTP Get interface (e.g., using key=value arguments), and a GLU-described interface.

Capability metadata, which describe what the service does, its limitations, and other behavioral characteristics.

4.4.7 Interface metadata

Service.AccessURL (URL)
Definition: The URL that a client uses to access a service.

Comment: The Service.AccessURL is the endpoint URL for the particular service. How this URL is to be interpreted and used depends on the particular type of service (e.g., REST vs. web service). If such interpretation and usage cannot be determined based on the service type, or if a service does not use standard defaults, Service.DefinitionURL and Service.BaseURL may be used for clarification.

Service.DefinitionURL (URL)
Definition: A URL pointing to a document that presents or describes the service inter-face.
Comment: For a Web Service, this would point to the WSDL document, for a GLU-described service, it would point to the GLU record, and for a browser-based service, this would be the Web page that actually contains the Web Form.

Service.BaseURL (URL)
Definition: The base portion of a URL used to invoke a service with the expectation that an additional string must be appended for the service to execute properly. The syntax of the appended string is defined by the specific service.

Service.HTTPResultsMIMEType (MIME type)
Definition: The MIME type that is returned by a service.

4.4.8 Capabilities metadata

Service.StandardID (URI)
Definition: An identifier for a standard service. The syntax for Identifiers is described in IVOA Identifiers in the IVOA document collection (http://www.ivoa.net/Documents/).
Comment: This provides a unique way to refer to a service specification standard, such as a Simple Image Access service. It assumes that such standard is registered and accessible. The Service.StandardID should include a specification of the version num-ber or release number of the given service.

Service.MaxReturnRecords (int)
Definition: Service providers may choose to restrict the number of records returned in order to avoid swamping the user with responses to an overly general query. If no value is provided, it is assumed that there is no restriction on the number of records returned.

Service.MaxReturnSize (float, bytes)
Definition: Service providers may choose to restrict the total size of a service response in order to manage computational resources. Service.MaxReturnSize allows providers to state such a restriction.

4.5 Example

Example: The Sloan Digital Sky Survey data as hosted by MAST at STScI (with no assertion that the metadata element values are actually correct, though they are not unreasonable).

Identity metadata

Title	*Sloan Digital Sky Survey*
ShortName	*SDSS*
Identifier	*ivo://stsci.edu/mast/sdss*

Curation metadata

Publisher	*Space Telescope Science Institute/MAST*
PublisherID	*ivo://stsci.edu/mast*
Creator	*Sloan Digital Sky Survey Consortium*
Creator.Logo	*http://archive.stsci.edu/images/sdss_logo.gif*
Contributor	*Sloan Digital Sky Survey Consortium*
Date	*2003-02-01*
Version	*SDSS EDR*
Contact.Name	*Archive Branch, Space Telescope Science Institute*
Contact.Address	*3700 San Martin Drive, Baltimore, MD 21218 USA*
Contact.Email	*archive@stsci.edu*
Contact.Telephone	*+1-410-338-4547*

General content metadata

Subject	*galaxies, quasars, stars, CCD photometry,spectroscopy, redshift, sky surveys*
Description	*The Sloan Digital Sky Survey is using a dedicated 2.5 m telescope and a large format CCD camera to obtain im-ages of over 10,000 square degrees of high Galactic latitude sky in five broad bands (u', g', r', i' and z', centered at 3540, 4770, 6230, 7630, and 9130 Å, respectively). Medium resolution spec-tra will be obtained for approximately 106 galaxies and 100,000 quasars. The early data release (EDR), on June 2001, includes searchable catalogs of images and spectra, images for display and scientific purpose in both 2-D FITS and JPEG formats, and spectra in both 1-D FITS and GIF formats. The EDR covers about 460 square degrees of sky. The next data releases will oc-cur every 18 months or so.*
Source	*2002AJ….123..485S*
ReferenceURL	*http://archive.stsci.edu/sdss/index.html*
Type	*Survey, Catalog, EPOResource*
ContentLevel	*Research*
Relationship	*mirror-of*
RelationshipID	*ivo://sdss.org/sdss/edr*

Collection and service content metadata

Facility	*Apache Point Observatory, Sloan 2.5-m Telescope*
Instrument	*Five-band clocked CCD camera*

Coverage.Spatial PositionInterval FK5 145.17 –1.25 235.9 1.25 PositionInterval FK5 250.71 52.15 267.0 66.29 PositionInterval FK5 350.43 –1.25 359.99 1.17 PositionInterval 0.0 –1.25 56.37 1.17

Coverage.RegionOfRegard	*0.0001*
Coverage.Spectral	*Optical*
Coverage.Spectral.Bandpass	*u', g', r', i', z'*
Coverage.Spectral.MinimumWavelength	*400.e-9*
Coverage.Spectral.MaximumWavelength	*850.e-9*
Coverage.Temporal.StartTime	*1999-12-25*
Coverage.Temporal.StopTime	*2001-07-15*
Coverage.Depth	*3.e-6*
Coverage.ObjectDensity	*6.e4*
Coverage.ObjectCount	*2.e7*
Coverage.SkyFraction	*0.01*
Resolution.Spatial	*0.00028*
Resolution.Spectral	*5000*
Resolution.Temporal	*120*
UCD	*Not Provided*
Format	*text/xml*
Rights	*Public*

Data quality metadata

DataQuality	*A*
ResourceValidationLevel	*4 [provided by registry curator]*
ResourceValidatedBy	*ivo:/us-vo.org/registry*
Uncertainty.Photometric	*3.e-7*
Uncertainty.Spatial	*0.00003*
Uncertainty.Spectral	*1.e-11*
Uncertainty.Temporal	*0.1*

Service metadata

Service.AccessURL	*http://archive.stsci.edu/cgi-bin/sdss/catalog*
Service.InterfaceURL	*http://archive.stsci.edu/sdss/catalog.html*
Service.BaseURL	*http://archive.stsci.edu/cgi-bin/sdss/catalog*
Service.HTTPResultsMIMEType	*text/xml*
Service.StandardID	*ivo://ivoa.net/Services/ConeSearch*
Service.MaxSearchRadius	*0.2*
Service.MaxReturnRecords	*5000*
Service.MaxReturnSize	*5.e8*"

5. VOTable Format Definition (Applications)

5.1 Introduction

The main goal of a virtual observatory is the exchange of data. Data are normally exchanged in the form of tables. "The VOTable format is an XML standard for the interchange of data represented as a set of tables. In this context, a table is an unordered set of rows, each of a uniform structure, as specified in the table description (the table metadata). Each row in a table is a sequence of table cells, and each of these contains either a primitive data type, or an array of such primitives" (Ochensenbein et al., 2013). VOTable ultimately originated in the FITS Table format. FITS (Flexible Image Transport System) is "much more than just another image format (such as JPG or GIF) and is primarily designed to store scientific data sets consisting of multidimensional arrays (images) and 2-dimensional tables organized into rows and columns of information" (please see http://fits.gsfc.nasa.gov/fits_primer.html). This section is an edited quotation from "VOTable Format Definition, Version 1.3, IVOA Recommendation 2013-09-20, found at http://www.ivoa.net/Documents/VOTable/20130920.

5.2 Actions for Tropical Forest Researchers

Tropical forest researchers can use a number of the generic VOTable standards for their own purposes, but they must also identify standards specific to their needs.

5.3 Overview

VOTable is designed as a flexible storage and exchange format for tabular data. Interoperability is encouraged through the use of standards (XML). The XML fabric allows applications to easily validate an input document, as well as facilitating transformations through XSLT (eXtensible Style Language Transformation) engines.

VOTable has built-in features for big-data and Grid computing. It allows metadata and data to be stored separately, with the remote data linked. Processes can then use metadata to 'get ready' for their input data, or to organize third-party or parallel transfers of the data. Remote data allow the metadata to be sent in email and referenced in documents without pulling the whole dataset with it: just as we are used to the idea of sending a pointer to a document (URL) in place of the document, so we can now send metadata-rich pointers to data tables in place of the tables themselves. The remote data is referenced with the URL syntax protocol://location, meaning that arbitrarily complex protocols are allowed.

The data part in a VOTable may be represented using one of four different formats: TABLEDATA, FITS, BINARY and BINARY2. TABLEDATA is a pure XML format so that small tables can be easily handled in their entirety by XML tools. The FITS binary table format is well-known to astronomers, and VOTable can be used either to encapsulate such a file, or to re-encode the metadata; unfortunately FITS requires a specification up front of the maximum size of its variable-length arrays. The BINARY and BINARY2 formats are supported for efficiency and ease of programming: no FITS library is required, and the streaming paradigm is supported.

VOTable can be used in different ways, as a data storage and transport format, and also as a way to store metadata alone (table structure only). In the latter case, a VOTable structure can be sent to a server, which can then open a high-bandwidth connection to receive the actual data, using the previously-digested structure as a way to interpret the stream of bytes from the data socket.

VOTable can be used for small numbers of small records (pure XML tables), or for large numbers of simple records (streaming data), or it can be used for small numbers of larger objects. In the latter case, there will be software to spread large data blocks among multiple processors on the Grid. Currently the most complex structure that can be in a VOTable Cell is a multidimensional array.

VOTable is constructed with XML (extensible Markup Language), a powerful standard for structured data throughout the Internet industries. It derives from SGML, a standard used in the publishing industry and for technical documentation for many years. XML consists of elements and payload, where an element consists of a start tag (the part in angle brackets), the payload, and an end tag (with angle brackets and a slash). Elements can contain other elements. Elements can also bear attributes (keyword-value combinations).

Following the general XML rule, element and attribute names are case-sensitive and have to be used with the specified capitalisation. For VOTable, we have adopted the convention that element names are spelled in uppercase and attribute names in lowercase (with an exception for the ID attribute).

5.4 Data Model

In this section we define the data model of a VOTable, and in the next sections its syntax when expressed as XML. The data model of VOTable can be expressed as:

VOTable = hierarchy of Metadata + associated TableData, arranged as a set of Tables

Metadata = Parameters + Infos + Descriptions + Links + Fields + Groups

Table = list of Fields + TableData

TableData = stream of Rows

Row = list of Cells

Cell = Primitive, or variable-length list of Primitives, or multidimensional array of Primitives

Primitive = integer, character, float, floatComplex, etc

Metadata is divided into that which concerns the table itself (parameters), and the definitions of the fields (or column attributes) of the table. Each FIELD represents the metadata that can be found at the top of the column in a paper version of the table:

A parameter (PARAM) is similar to a FIELD, except that it has a value attribute. Parameters can be seen as "constant columns", containing for instance FITS keywords or any other information pertaining to the table itself or its environment.

An informative parameter (INFO) is a restricted form of the PARAM — it is always understood as a string (i.e. datatype="char" and arraysize="" are implied).*

The ordered list of Fields at the top of the table thus provides a template for a Row object (also called a record). The template allows interpretation of the data in the Row. The record is a set of Cells, with the number and order of Cells the same for each Row, and the same as the number of Fields defined in the Metadata.

In VOTable, there is generally no advance specification of the number of rows in the table: this is to allow streaming of large tables, as discussed above. However, if the number of rows is known, it may be specified in a dedicated nrows attribute.

Columns may be logically grouped, so that it is possible to define table substructures made of column associations. Such an association is declared as a GROUP, which typically contains column references (FIELDref) and associated parameters (PARAM).

Each Cell is composed from Primitives, each of which is a datatype of fixed-length binary representation. Cells may consist of a single Primitive (this is the default), or of an array (which may be multidimensional) of Primitives.

Except for the Bit type, each primitive has the fixed length in bytes. Bit scalars and arrays are stored in the minimum number of bytes feasible.

VOTables support two kinds of characters: ASCII 1-byte characters and Unicode (UCS-2) 2-byte characters. Unicode is a way to represent characters that is an

alternative to ASCII. It uses two bytes per character instead of one, it is strongly supported by XML tools, and it can handle a large variety of international alphabets. Therefore VOTable supports not only ASCII strings (datatype="char"), but also Unicode (datatype="unicodeChar"). Note that strings are not a primitive type: strings are represented in VOTable as an array of characters.

A table cell can contain an array of a given primitive type, with a fixed or variable number of elements; the array may even be multidimensional. A table cell can also contain a multidimensional array of a given primitive type. This is specified by a sequence of dimensions separated by the x character, with the first dimension changing fastest.

Strings, which are defined as a set of characters, can therefore be represented in VOTable as a fixed- or variablelength array of characters.

VOTable is closely compatible with the FITS Binary Table format. Henceforth, we shall abbreviate "FITS Binary Table and its Conventions" simply by the word "FITS". Given a FITS file that represents a binary table, the header may be converted to VOTable, with a pointer to the original file, or with the original file included directly in VOTable. Since the original file is still present, it is clear that no data has been lost. A PARAM element can be used to hold any FITS keyword with its value and comment string.

What can FITS do but not VOTable? FITS has complex semantics, with many conventions (see e.g. the Registry of FITS Conventions) which have been developed mainly to be able to cope with the increasing complexity of astronomical instrumentation. In the frame of the Virtual Observatory the complexity is described by means of data models, and VOTable can refer to these data models by means of the utype attribute.

What can VOTable do but not FITS? VOTable supports separating of data from metadata and the streaming of tables, and other ideas from modern distributed computing. It bridges two ways to express structured data: XML and FITS. It uses UCDs to formally express the semantic content of a parameter or field. It has the hierarchy and flexibility of XML: using GROUP elements introduced in version 1.1, columns in a VOTable can be grouped in arbitrarily complex hierarchies; and the ID attribute can be used in XML to enable what are essentially pointers. FITS does not handle Unicode (extended alphabet) characters.

It should be noticed that the transformation of FITS to VOTable is reversible: any FITS table can be converted to a VOTable without loss of information and the resulting VOTable can be converted back to a FITS table also without loss of information. However, it is possible to create new VOTables which cannot be converted to FITS tables without loss of information.

5.5 The VOTable Document Structure

The overall VOTable document structure is described and controlled by its XML Schema. That means that documents claiming to represent VOTables must include the reference to the VOTable schema, and pass through W3C XML Schema validators

without error; notice that the validation is a necessary, but not sufficient, condition for correctness.

A VOTable document consists of a single all-containing element called VOTABLE, which contains descriptive elements and global definitions (DESCRIPTION, GROUP, PARAM, INFO), followed by one or more RESOURCE elements. Each Resource element contains zero or more TABLE elements, and possibly other RESOURCE elements. The TABLE element, the actual heart of VOTable, contains a description of the columns and parameters followed by the data values.

5.6 Example

This simple example of a VOTable document lists 3 galaxies with their position, velocity and error, and their estimated distance.

```
<?xml version="1.0"?>
<VOTABLE version="1.3" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.ivoa.net/xml/VOTable/v1.3"
xmlns:stc="http://www.ivoa.net/xml/STC/v1.30" >
<RESOURCE name="myFavouriteGalaxies">
<TABLE name="results">
<DESCRIPTION>Velocities and Distance estimations</DESCRIPTION>
<GROUP utype="stc:CatalogEntryLocation">
<PARAM name="href" datatype="char" arraysize="*"
utype="stc:AstroCoordSystem.href" value="ivo://STClib/CoordSys#UTC-ICRS-TOPO"/>
<PARAM name="URI" datatype="char" arraysize="*"
utype="stc:DataModel.URI" value="http://www.ivoa.net/xml/STC/stc-v1.30.xsd"/>
<FIELDref utype="stc:AstroCoords.Position2D.Value2.C1" ref="col1"/>
<FIELDref utype="stc:AstroCoords.Position2D.Value2.C2" ref="col2"/>
</GROUP>
<PARAM name="Telescope" datatype="float" ucd="phys.size;instr.tel"
unit="m" value="3.6"/>
<FIELD name="RA" ID="col1" ucd="pos.eq.ra;meta.main"
datatype="float" width="6" precision="2" unit="deg"/>
<FIELD name="Dec" ID="col2" ucd="pos.eq.dec;meta.main"
datatype="float" width="6" precision="2" unit="deg"/>
<FIELD name="Name" ID="col3" ucd="meta.id;meta.main"
datatype="char" arraysize="8*"/>
<FIELD name="RVel" ID="col4" ucd="spect.dopplerVeloc" datatype="int"
width="5" unit="km/s"/>
<FIELD name="e_RVel" ID="col5" ucd="stat.error;spect.dopplerVeloc"
datatype="int" width="3" unit="km/s"/>
<FIELD name="R" ID="col6" ucd="pos.distance;pos.heliocentric"
datatype="float" width="4" precision="1" unit="Mpc">
<DESCRIPTION>Distance of Galaxy, assuming H=75km/s/Mpc</DESCRIPTION>
</FIELD>
<DATA>
<TABLEDATA>
```

<TR>
<TD>010.68</TD><TD>+41.27</TD><TD>N224</TD><TD>297</TD><TD>
5</TD><TD>0.7</TD>
</TR>
<TR>
<TD>287.43</TD><TD>-63.85</TD><TD>N6744</TD><TD>839</TD><TD>
6</ TD><TD>10.4</TD>
</TR>
<TR>
<TD>023.48</TD><TD>+30.66</TD><TD>N598</TD><TD>182</TD><TD>
3</ TD><TD>0.7</TD>
</TR>
</TABLEDATA>
</DATA>
</TABLE>
</RESOURCE>
</VOTABLE>

This simple VOTABLE document shows a single RESOURCE made of a single TABLE; the table is made of 6 columns, each described by a FIELD, and has one additional PARAM parameter (the Telescope). The actual rows are listed in the DATA part of the table, here in XML format (introduced by TABLEDATA); each cell is marked by the TD element, and follow the same order as their FIELD description: RA, Dec, Name, RVel, e RVel, R.

5.7 Other Elements

5.7.1 name, ID and ref attributes

Most of the elements defined by VOTable may have or have to have names, like a RESOURCE, a TABLE, a PARAM or a FIELD. The content of the name attribute is defined as a token XML type, that is a string of characters where the blanks and spaces are not meaningful (no leading or trailing spaces, no multiple spaces): name="NVSS flux(1.4GHz)" represents therefore a valid name.

The ID and ref attributes are defined as XML types ID and IDREF respectively. This means that the content of ID is an identifier which must be unique throughout a VOTable document, and that the content of the ref attribute represents a reference to an identifier which must exist in the VOTable document.

5.7.2 VOTABLE element

The VOTABLE element may contain definitions consisting of a DESCRIPTION, followed by any mixture of parameters and informative notes eventually structured in groups. These elements represent values which are meaningful over all tables included in a VOTABLE document — definitions specific to a RESOURCE (section 3.4) or a TABLE (section 3.6) are better placed within their most appropriate element.

5.7.3 RESOURCE element

A VOTable document contains one or more RESOURCE elements, each of these providing a description and the datavalues of some logically independent data structure.

Each RESOURCE may include the descriptive element DESCRIPTION, followed by a mixture of INFO, GROUP and PARAM elements; it may also contain LINK elements to provide URL-type pointers that give further information.

The main component of a RESOURCE is typically one or more TABLE elements – in other words a RESOURCE is basically a set of related tables. The RESOURCE is recursive (it can contain other RESOURCE elements), which means that the set of tables making up a RESOURCE may become a tree structure.

A RESOURCE may have one or both of the name or ID attributes (see section 3.2); it may also be qualified by type="meta", meaning that the resource is descriptive only, i.e. does not contain any actual data: no DATA element should exist in any of its sub-elements. A RESOURCE without this attribute may however have no DATA sub-element.

5.7.4 TABLE Element

The TABLE element represents the basic data structure in VOTable; it comprises a description of the table structure (the metadata) essentially in the form of PARAM and FIELD elements, followed by the values of described fields in a DATA element.

The TABLE element is always contained in a RESOURCE element: in other words any TABLE element has a single parent made of the RESOURCE element in which the table is embedded.

The TABLE element contains a DESCRIPTION element for descriptive remarks, followed by a mixed collection of PARAM, FIELD or GROUP elements which describe a parameter (constant column), a field (column) or a group of columns respectively.

Furthermore the TABLE element may contain LINK elements that provide URL-type pointers, exactly like the LINK elements existing within a RESOURCE element.

The last element included in a TABLE is the optional DATA element: a table without any actual data is quite valid, and is typically used to supply a complete description of an existing resource e.g. for query purposes.

6. Simple Search (Data Access Layer)

6.1 Introduction

The main aim of a virtual observatory is to enable users to obtain data to meet their needs. This requires mechanisms which enable users to search for the data they require, and enable those who wish to make data available to do so in a way that is

compatible with the search mechanism. The very first standard that was developed for AstroGrid was the Simple Cone Search, so-called because it defines the area of interest in the sky in terms of two positional indicators and an angular indicator which combine to give a cone. In a Pan-Tropical Virtual Observatory this could be translated into a rectangle described by latitudinal and longitudinal coordinates. This section directly quotes from "Simple Cone Search, Version 1.03, IVOA Recommendation 22 February 2008". which may be found at http://www.ivoa.net/Documents/REC/DAL/ConeSearch-20080222.html.

6.2 Actions for Tropical Forest Researchers

The Simple Cone Search standard does contain generic elements, but to make the Simple ASTROTROP Search operationally feasible, tropical forest researchers must identify the search terms which they will need to use every day, to locate information layers on forest area, biodiversity, carbon and other attributes. Our needs will be far more demanding than the "RA, DEC and SR" search terms used by astronomers.

6.3 Overview

"This specification defines a simple query protocol for retrieving records from a catalog of astronomical sources. The query describes sky position and an angular distance, defining a cone on the sky. The response returns a list of astronomical sources from the catalog whose positions lie within the cone, formatted as a VOTable.

This specification describes how a provider of an astronomical source catalog can publish that catalog to the Virtual Observatory in such a way that a simple cone search can be done. The data remains in the control of the data provider, served through a web server to the world, but the query profile and response profile are carefully controlled, as described below. It is intended that setting up this web service be simple enough that data providers will not have to spend too much time on it (their funding to support such services is typically small). At the same time, the service implementation and the data it provides can serve as a basis for more sophisticated tools.

This specification does not specify how Cone Search services are implemented, or how the data are stored or manipulated. The concern of this specification is how the data are exposed to the world through well-defined requests and responses. This specification assumes that the data provider has already selected a catalog of astronomical sources. This catalog can be presented as a single table; it is expected that the table contains several columns.

6.3 Service Interface Requirements

A service implementation that is compliant with this standard must meet the following requirements:

1. the service must respond to a HTTP GET request represented by a URL having two parts:

a. A base URL of the form:

http://<server-address>/<path>?[<extra-GET-arg>&[...]]

where <server-address> and <path> are URI-compliant components indicating the domain address and local location path where the service is deployed. The <server-address> may end with one or more locally supported URL arguments, <extra-GET-arg>; these arguments are not recognized parts of the Cone Search protocol and thus are treated opaquely by clients of the service as part of the base URL.

Note that when it contains extra GET arguments, the base URL ends in an ampersand, &; if there are no extra arguments, then it ends in a question mark?

Every query to a given cone search service uses the same base URL.

b. Constraints expressed as a list of ampersand-delimited GET arguments of the form:
<name>=<value>

where <name> is a parameter name specified by this specification and <value> is its value. The constraints represent the query parameters which can vary for each successive query. The order of the name-parameter pairs is not significant.

The baseURL and constraint list are concatenated to form the query.

The set of query constraints must include the following parameters, which are interpreted by the service with the stated meaning:

a. RA - a right-ascension in the ICRS coordinate system for the positon of the center of the cone to search, given in decimal degrees.
b. DEC - a declination in the ICRS coordinate system for the positon of the center of the cone to search, given in decimal degrees.
c. SR - the radius of the cone to search, given in decimal degrees.

Example:
http://mycone.org/cgi-bin/search?RA=180.567&DEC=-30.45&SR=0.0125

The service must respond with an XML document in the VOTable format, that represents a table of astronomical sources whose positions are within the cone.

The VOTable MUST comply with these conditions:

a. There must be a single RESOURCE in the VOTable, and that contains a single TABLE.
b. The TABLE must have FIELDS where the following UCD [UCD] values have been set. There must only be one FIELD with each of these UCDs:
c. Exactly one FIELD must have ucd="ID_MAIN", with an array character type (fixed or variable length), representing an ID string for that record of the table. This identifier may not be repeated in the table, and it could be used to retrieve that same record again from that same table.
d. Exactly one FIELD must have ucd="POS_EQ_RA_MAIN", with type double, representing the right-ascension of the source in the ICRS coordinate system.

e. Exactly one FIELD must have ucd="POS_EQ_DEC_MAIN", with type double, representing the declination of the source in the ICRS coordinate system.
f. The VOTable may include an expression of the uncertainty of the positions given in the above mentioned fields to be interpreted either as a positional error of the source positions, or an angular size if the sources are resolved.

3. The service must respond with a stubbed version of the VOTable in the case of error. This must happen if the three required parameters (RA, DEC, SR) are not all present, or if their values cannot be parsed to floating-point numbers, or if the numbers are out of range (DEC=91.0, for example). In the case of error, the service MUST respond with a VOTable that contains a single PARAM element or a single INFO element with name="Error", where the corresponding value attribute should contain some explanation of the nature of the error.

6.4 The Resource Profile

A Cone Search service MUST be described with a Resource Profile that includes the following information. The Profile is composed of named metadata listed below. The format used to encode the profile should be compliant with the publishing standards of the IVOA throughout the time the service is supported by the data provider. The metadata names and values used in the profile encoding should match those given below as closely as possible; where they do not match exactly, they should be consistent with the IVOA metadata conventions in place at any given time and the mapping of names and values actually used and those given below should be well documented.

Several of the metadata listed below can have values that are hierarchical; this hierarchy should be represented in a manner most appropriate to the format used. When the format does not provide any such mechanism, it is recommended that the value be represented as a strings delimited by dots with the root domain of the value appearing first.

The resource profile consists of the following metadata with the stated definitions:

a. ResponsibleParty: The data provider's name and email.
b. ServiceName: The name of the catalog served by the service, for example "IRSA.2MASS.ExtendedSources".
c. Description: A couple of paragraphs of text that describe the nature of the catalog and its wider context.
d. Instrument: The instrument that made the observations, for example STScI.HST.WFPC2.
e. Waveband: The waveband of the observations, with ONE selected from this list: radio, millimeter, infrared, optical, ultraviolet, xray, gammaray.
f. Epoch: The epoch of the observations, as a free-form string.
g. Coverage: The coverage on the sky, as a free-form string.
h. MaxSR: The largest search radius, given in decimal degrees, that will be accepted by the service without returning an error condition. A value of 180.0 indicates that there is no restriction.
i. MaxRecords: The largest number of records that the service will return.

j. Verbosity: True or false, depending on whether the service supports the VERB keyword in the request.
k. BaseURL: The base URL for the service as described in Section 2.

The service will be considered published to the VO if the profile has been added to an IVOA Registry according to the IVOA standards and conventions at the time the service is made available, TOGETHER with maintaining the web service that is described by the profile in compliant order."

7. Registry Interfaces (Registry)

7.1 Introduction

Registries are the "yellow pages" where users search for databases that offer them the data which they need. Effective registries are therefore essential for effective virtual observatories. According to the IVOA Standard on Registry Interfaces, "Registries provide a mechanism with which VO applications can discover and select resources—e.g. data and services—that are relevant for a particular scientific problem. This specification defines the interfaces that support interactions between applications and registries as well as between the registries themselves. It is based on a general, distributed model composed of so-called *searchable* and *publishing* registries. The specification has two main components: an interface for searching and an interface for *harvesting*" (Benson et al., 2009). This section reproduces relevant parts of "IVOA Registry Interfaces, Version 1.0. IVOA Recommendation 2009 November 4", which is found at http://www.ivoa.net/Documents/RegistryInterface/20091104/, as a direct quote from Benson et al. (2009) indicated by italics.

7.2 Actions for Tropical Forest Researchers

This standard enables VOs to design their own registry in a way that enables communication with existing registries. Consequently, tropical forest researchers both design their own registry and identify existing registries that will be of value to their virtual observatory. As in the case of vocabularies this prevents the need to "reinvent the wheel", by duplicating previous work, and therefore will save time.

7.3 Overview

"In the Virtual Observatory (VO), registries provide a means for discovering useful data and services. To make discovery efficient, a registry typically represents to some extent a centralized warehouse of resource descriptions; however, the source of this information—the resources themselves and the data providers that maintain them—are distributed. Furthermore, there need not be a single registry that serves the entire international VO community. Given the inherent distributed nature of the information used for resource discovery, there is clearly a need for common mechanisms for registry communication and interaction.

This document describes the standard interfaces that enable interoperable registries. These interfaces are based in large part on a Web Service definition in the form of a WSDL document, which is included in this specification. Through these interfaces, registry builders have a common way of sharing resource descriptions with users,

applications, and other registries. Client applications can be built according to this specification and can discover and retrieve descriptions from any compliant registry.

This specification does not preclude a registry builder from providing additional value-added interfaces and capabilities. In particular, they are free to build interactive, end-user interfaces in any way that best serves their target community. In a similar spirit, this specification does not intend to enforce completely identical behavior of required operations across all compliant implementations. In particular, this specification does not require that identical search queries sent to different compliant registries return identical results. Implementations are free to support different strategies for evaluating an ambiguous query (such as a keyword search) and ordering the results in a way that best serves the target community.

7.4 Registry Architecture and Definitions

A registry is first a repository of structured descriptions of resources. In the VO, a resource is defined by the IVOA Recommendation, "Resource Metadata for the Virtual Observatory" (RM) [Hanisch 2004] as : "A resource is a general term referring to a VO element that can be described in terms of who curates or maintains it and which can be given a name and a unique identifier. Just about anything can be a resource: it can be an abstract idea, such as sky coverage or an instrumental setup, or it can be fairly concrete, like an organization or a data collection."

Organizations, data collections, and services can be considered as classes of resources. The most important type of resource to applications is a service that actually does something. What is available at a particular resource is described through the content of metadata, whereas the service metadata describes how to access it. The RM describes a registry, then, as "a service for which the response is a structured description of resources" [Hanisch 2004]. Each resource description it returns is referred to as a resource record.

In the registry model, the VO environment features different types of registries that serve different functions. The primary distinction is between publishing registries and searchable ones.

A searchable registry is one that allows users and client applications to search for resource records using selection criteria against the metadata contained in the records. The purpose of this type of registry is to aggregate descriptions of many resources distributed across the network. By providing a single place to locate data and services, applications are saved from having to visit many different sites to just to determine which ones are relevant to the scientific problem at hand. A searchable registry gathers its descriptions from across the network through a process called harvesting.

A publishing registry is one that simply exposes its resource descriptions to the VO environment in a way that allows those descriptions to be harvested. The contents of these registries tend to be limited to resources maintained by one or a few providers and thus are local in nature; for example, a data center will run its own publishing registry to expose all the resources it maintains to the VO environment. Since the purpose is simply publishing and not to serve users and applications directly, it is not

necessary to support full searching capabilities. This simplifies the requirements for a publishing registry: not only does it not need to support the general search interface, the storage and management of the records can be simpler. While a searchable registry in practice will necessitate the use of a database system, a publishing registry can easily store its records as flat files on disk.

Note that some registries can play both roles; that is, a searchable registry may also publish its own resource descriptions.

A secondary distinction is full versus local registries. A full registry is one that attempts to contain records of all resources known to the VO. In practice, this attribute is associated only with searchable registries, as in the so-called full searchable registry. It is expected that there will be several such registries available, perhaps each run by a major VO project; this not only avoids the single point of failure, but also allows some specialization to serve the particular needs of the project that maintains it. A local registry, on the other hand, contains only a subset of known resources. In practice, all publishing registries are local; however, we expect that there may be local searchable registries that specialize in particular types of resources, perhaps oriented toward a scientific topic.

As mentioned above, harvesting is the mechanism by which a registry can collect resource records from other registries. This mechanism is used by full searchable registries to aggregate resource records from many publishing registries. It can also be used to synchronize two registries to ensure that they have the same contents. Harvesting, in this specification, is modeled as a "pull" operation between two registries. The harvester refers to the registry that wishes to receive records (usually a searchable registry); it sends its request to the harvestee (usually the publishing registry), which responds with the records. Harvesting is intended to be a much simpler process than search and retrieval; nevertheless, there are at least two kinds of filtering that a harvestee needs to support:
a. Filtering by date: this allows the harvester to return to the harvestee periodically to retrieve only new and updated records.

b. Filtering by ownership: harvesting only those records that originated with the harvestee (as opposed to those that may have been harvested from other registries) prevents a harvester from receiving duplicate records from multiple registries.

7.5 Specification Summary

The purpose of the registry is to be used by other applications to provide access to various types of resources. At the programmatic level, connectivity between the registry and other applications is ensured through the registry interface as defined by this document.

The registry interface is composed of two independent parts. The harvesting interface provides the mechanism for registries to talk to each other and share information. The searching interface is used by clients that want to discover resources to use as part of a VO application. A registry may implement either interface or both, depending on the roles it intends to play.

The searching interface can return XML descriptions of resources, or resource records, and it consists of four required operations:

a. Search returns active resource records that match a specific set of constraints.
b. KeywordSearch returns active resource records containing specified keywords.
c, GetResource returns a single resource records identified by its unique IVOA identifier.
d. GetIdentity returns the resource record describing the searchable registry itself.

The harvesting interface, which allows one registry to collect resource records from other registries, leverages the OAI-PMH standard. The OAI-PMH interface is composed of six operations. The most commonly used harvesting operation is the ListRecords, which can return the descriptions of the resources of a particular category, or set, that have been created or updated since a specified date. Normally the harvester will request records from the set that originate from that registry (as opposed to those harvested from another registry). This set is said to be managed by the registry. The complete list of OAI-PMH harvesting operations is:

a. Identify returns the resource record describing the harvestable registry itself.
b. ListIdentifiers lists the identifiers of records that have changed since a given date
c. ListRecords lists the full descriptions of resources that have changed since a given date.
d. GetRecord returns a single record identified by its identifier.
e. ListMetadataFormats lists the available output description formats.
f. ListSets lists the categories of records that can be requested.

The searching and harvesting operations that return resource descriptions do so using the VOResource XML Schema and any of its legal extensions [VOResource].

7.6 Searching

The four required operations that make up the searching interface fall into two groups: search and resolve. The search operations—Search and KeywordSearch—return a list of one or more resource descriptions held by the registry that matches the input selection criteria. The Search operation supports constraint-based searching for resources by means of a query using the Astronomical Data Query Language (ADQL), KeywordSearch provides keyword-based searching for resources whose descriptions contain words in an input string.

The resolve operations - GetResource and GetIdentity - each return one and only one resource description. The GetResource resolves a unique IVOA Identifier [Identifier] to the description of the associated resource. The GetIdentity returns the resource record of the searchable registry itself.

These operations are implemented to accept input parameters and return output results as SOAP messages.

7.7 Required Search Operations

The two search operations—Search and KeywordSearch—return resource records that match a set of selection criteria.

7.7.1 Output Format

The two search operations share a common output format for the resource records that match the search criteria. The response is a SOAP message. This message is defined to have a single part: a SearchResponse element from the http://www.ivoa.net/wsdl/RegistrySearch/v1.0 namespace. This element in turn wraps a single VOResources element from the http://www.ivoa.net/xml/RegistryInterface/v1.0 namespace (from now on referred to using the "ri:" prefix) that contains each of the matching records and conforms to the following XML Schema definition:

```
VOResources and Resource Element Definitions
<xs:element name="VOResources">
  <xs:complexType>
    <xs:sequence>
      <xs:choice>
        <xs:element ref="ri:Resource"
                minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="identifier" type="vr:IdentifierURI"
                minOccurs="0" maxOccurs="unbounded"/>
      </xs:choice>
    </xs:sequence>
    <xs:attribute name="from" type="xs:positiveInteger"
            use="required" />
    <xs:attribute name="numberReturned" type="xs:positiveInteger"
            use="required" />
    <xs:attribute name="more" type="xs:boolean" use="required" />
  </xs:complexType>
</xs:element>

<xs:element name="Resource" type="vr:Resource"/>
```

The required ri:VOResources attributes allow the results of the query to be "paged" over several calls that can be controlled via the operation input parameters, from and max. They assume that over some limited amount of time multiple calls to a search operation on a single registry with the same search constraints returns the same results and in the same order. If the client does not request paging via the from and max parameters, the service may choose to return partial results (setting the appropriate attributes) if the results exceed the service's own internal limits.

The contents of the ri:VOResources element depends on the value of the operation input parameter, identifiersOnly. If this parameter is set to true, then the ri:VOResources element must contain a list of identifier elements that contain the IVOA identifiers of resources that match the input query. If identifiersOnly is false, then the ri:VOResources element must contain a list of ri:Resource elements containing the full VOResource descriptions of resources that matches the query.

The search responses must include only active resource records—that is, records in which the vr:Resource element's status attribute is set to the value, "active". Thus, clients do not have to explicitly in include a search constraint to avoid deleted or inactive records. (GetResource and XQuerySearch operations, on the other hand, may return inactive or deleted resources.)

The search responses must include the xsi:schemaLocation attribute (regardless of the value of identifiersOnly) in compliance with the XML Schema standard [Schema] to indicate a URL location for the VOResource schema and all of the legal extensions of VOResource that are employed in the response. This xsi:schemaLocation attribute must appear either as an attribute of the ri:VOResources element or as an attribute of each child ri:Resource element (when identifiersOnly is false) or both.

If a legal search query does not match any resource records, the ri:VOResources element must contain no ri:Resource elements. If the input search query is illegal in its syntax or the operation encounters any other error that prevents returning the requested records, the operation must return an ErrorResponse fault, represented by an ErrorResponse element.

7.7.2 Constraint-based Search Query

The Search operation allows clients to retrieve a list of resource descriptions that match constraints of values corresponding to specific metadata from the VOResource schema (and its legal extensions). The operation's input message is defined to have a single part, a Search element, which contains four child elements that serve as the four input parameters: where, from, max, and identifiersonly.

Search Element Definition

```
<xs:element name="Search">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="tns:Where" minOccurs="1" maxOccurs="1" />
      <xs:element name="from" type="xs:positiveInteger"
            minOccurs="0" maxOccurs="1" />
      <xs:element name="max" type="xs:positiveInteger"
            minOccurs="0" maxOccurs="1" />
      <xs:element name="identifiersOnly" type="xs:boolean"
            minOccurs="0" maxOccurs="1" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

The one required parameter, the Where element, is of type whereType from the ADQL XML Schema (having the namespace, http://www.ivoa.net/xml/ADQL/v1.0, from now on referred to using the "adql:" prefix) which contains the constraints that specific components of the resource metadata must satisfy.

The specific components are named within search constraints (represented by adql:Condition elements) using adql:Arg elements subject to the following restrictions:

a. The Table attribute, which is required by the ADQL Schema, should be set to an empty string and must be ignored by the Search method implementation.

b. The Name attribute, which is required by the ADQL Schema, may be set to an empty string or to a short name to serve as an alias for the resource metadata referred to. This value must be ignored by the Search method.

c. The xpathName attribute must be set to a restricted XPath string. This XPath string identifies the specific VOResource element (or legal extension) within the resource record that is to be constrained.

7.7.3 Keyword Search Query

The purpose of the KeywordSearch operation is to provide a simple way to select resources based on the string values in their resource descriptions. The operation only queries for active resources noted by status='active'. The operation's input message is defined to have a single part, a KeywordSearch element, which contains five child elements that serve as the five input parameters:

```
KeywordSearch Element Definition
<xs:element name="KeywordSearch">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="keywords" type="xs:string"
            minOccurs="1" maxOccurs="1" />
      <xs:element name="orValues" type="xs:boolean"
            minOccurs="0" maxOccurs="1" default="true"/>
      <xs:element name="from" type="xs:positiveInteger"
            minOccurs="0" maxOccurs="1" />
      <xs:element name="max" type="xs:positiveInteger"
            minOccurs="0" maxOccurs="1" />
      <xs:element name="identifiersOnly" type="xs:boolean"
            minOccurs="0" maxOccurs="1" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

The meaning of the last three parameters above and their effect on the output is the same as for the Search operation.

The keywords parameter is a string that consists of one or more words separated by whitespace characters. The characters that qualify as whitespace are the same as in XML: space (x20), tab (x9), line feed (xA), and carriage return (xD). A phrase is a portion of keywords parameter that is enclosed in double quotation marks (e.g. "black hole"). Words and phrases are extracted from the keywords parameter as a list of tokens to be used in the search. When a phrase is extracted from the parameter, the quotation marks are removed.

The KeywordSearch implementation forms a query by, in effect, creating a search constraint for each word or phrase in this parameter. For each active or inactive resource record, each word or phrase is compared against every value for a selected

set of resource metadata that includes at minimum the following (drawn from the VOResource schema):

a. identifier: the resource's IVOA identifier
b. content/description: the descriptive summary of the resource
c. title: the resource title
d. @xsi:type: the specific type of resource specified as an extension of the Resource type
e. content/subject: the subject topics associated with the resource
f. content/type: the general type of resource

7.8 Resolve Operations

The two resolve operations—GetResource and GetIdentity—each select and return a single resource record.

The two resolve operations share a common output format for returning a single resource record. The response is a SOAP message. This message is defined to have a single part: a ResolveResponse element from the http://www.ivoa.net/wsdl/RegistrySearch/v1.0 namespace. This element in turn wraps an ri:Resource element of type vr:Resource or one of its legal extensions.

The Resource element must include a xsi:schemaLocation attribute in compliance with the XML Schema standard [Schema] to indicate a URL location for the VOResource schema and all of the legal extensions of VOResource that are employed in the response.

The purpose of the GetResource operation is to provide a simple way to resolve a unique IVOA Identifier to a full resource description. The input message is defined to have a single part, a GetResource element. This element contains the operation's one input parameter, identifier, of type vr:IdentifierURI, encodes an IVOA identifier. The output message contains a single VOResource record whose identifier element matches the input identifier.

The purpose of the GetIdentity operation is to provide a simple way to get the VOResource record that describes the implementing registry itself. A client may then inspect this VOResource record to discover various information about the implemented registry.

7.9 Harvesting Interface

Harvesting is the mechanism by which a registry can collect resource descriptions from other registries. This mechanism is used by full searchable registries to aggregate resource descriptions from many publishing registries. It can also be used to synchronize two registries to ensure that they have the same contents. This section defines the IVOA Harvesting Interface. Client applications that make use of this interface are referred to as harvesters. Those registries that declare themselves as harvestable must comply with the specification described in this section.

This specification defines two variants of the harvesting interface, both built on the standard Protocol for Metadata Harvesting developed by Open Archives Initiative (OAI-PMH). The first variant is one that is fully compliant with the OAI-PMH version 2.0 standard; harvestable registries must support this variant. Compliance with this base standard allows IVOA registries to be accessed by applications from outside the IVOA community. The second variant, a Web Service version of the OAI interface (in which the input and outputs are transported as SOAP messages), is also defined as an optional alternative.

The required variant of the interface is defined by the OAI-PMH v2.0 specification, which itself defines:

a. the meaning and behavior of the six harvesting operations, referred to as "verbs",
b. the meaning of the input arguments for each operation, and
c. the XML Schema used to encode response messages.

In summary, the OAI-PMH standard defines six operations:

a. Identify: provides a description of the registry
b. ListIdentifiers: returns a list of identifiers for the resource records held by the registry.
c. ListRecords: returns all Resource records in the registry. Registries may use the set "ivo_managed" to get Resource records managed by this particular registry.
d. GetRecord: returns a single resource description matching a given identifier.
e. ListMetadataFormats: returns a list of supported formats that the registry can use to encode resource descriptions upon a harvester's request.
f. ListSets: return a list of category names supported by the registry that harvesters can request in order to get back a subset of the descriptions held by the registry.

The ListRecords and GetRecord operations return the actual resource description records held by the registry. These descriptions are encoded in XML and wrapped in a general-purpose envelope defined by the OAI-PMH XML Schema.

Through the operations' arguments, OAI-PMH provides a number of useful features:

a. Support for multiple return formats. As suggested by the ListMetadataFormats operation, a harvester can request the formats available for encoding returned resource descriptions.

b. Harvesting by date. The ListIdentifiers and ListRecords operations both support "from" and "until" date arguments. The "from" argument can be used to retrieve records that have changed since the last harvest.

c. Harvesting by category. The ListIdentifiers and ListRecords operations both support a "set" argument for retrieving resources that are grouped in a particular category. Resource records may belong to multiple groups.

d. Marking records as deleted. Registries may mark records as deleted so that harvesters may remove access to them from their applications. Registries may permanently remove deleted resources that have been marked deleted for more than six months.

e. Support for resumption tokens. If a request results in returning a very large number of records, the registry can choose to split the results over several calls; this is done by passing a resumption token back to the harvester. The harvester uses it to retrieve the next set of matching results.

f. Harvesting with no date. Deleted resource records may not be returned when no "from" or "until" is specified.

7.10 Metadata Formats for Resource Descriptions

All IVOA registries that support the Harvesting Interface must support two standard metadata formats: the OAI Dublin Core format (mandated by the base OAI-PMH standard) and the IVOA VOResource metadata format.

The VOResource metadata format will have the metadata prefix name "ivo_vor" which can be used wherever an OAI-PMH metadata prefix name is supported. The format uses the VOResource core XML Schema with the namespace http://www.ivoa.net/xml/VOResource/v1.0 (referred hereto with the namespace prefix "vr") along with any legal extension of this schema to encode the resource descriptions within the OAI-PMH metadata tag from the OAI XML Schema (namespace http://www.openarchives.org/OAI/2.0, hereto referred by the namespace prefix "oai"). The format is specifically represented by an element called Resource from the http://www.ivoa.net/xml/RegistryInterface/v1.0 namespace (from now on referred to using the "ri:" prefix) as the sole child of the oai:metadata element. The ri:Resource element must include an xsi:type attribute that assigns the element's type to vr:Resource or one of its legal extensions.

The OAI Dublin Core format, with the metadata prefix of "oai_dc", is defined by the OAI-PMH base standard and must be supported by all OAI-PMH compliant registries. This document does not specify how a record in the VOResource format maps into the OAI Dublin Core format; however, the IVOA Registry Working Group may recommend such a mapping based on the IVOA Resource Metadata standard.

In accordance with the OAI-PMH standard, an OAI-PMH XML envelope that contains a resource description must include a globally unique URI that identifies that resource record. This identifier must be the IVOA identifier used to identify the resource being described and cited as the value of the vr:identifier resource metadata.

7.11 Required Records

This section describes the records that a harvestable IVOA Registry must include among those it emits via the OAI-PMH operations.

The harvestable registry must return one record that describes the registry itself as a whole, and the "ivo_vor" format must be supported for this record. This record is included in the Identify operation response. When encoded using the "ivo_vor" format, the returned ri:Resource element must be of the type Registry from the VORegistry schema (namespace http://www.ivoa.net/xml/VORegistry/v1.0; hereto referred by the "vg" namespace prefix). The record must include a vg:managedAuthority for every Authority Identifier [Identifiers] that originated at that registry. The registry may contain other registry records for other registries it knows about; use of a vr:Resource extension type other than vg:Registry to describe these other registries is strongly discouraged.

7.12 The Identify Operation

The Identify operation describes the harvestable registry as a whole. The response from this operation must include all information required by the OAI-PMH standard. In particular, it must include an oai:baseURL element that must refer to the base URL to the harvesting interface endpoint. When the Identify operation is called through the Web Service variant, the oai:baseURL element value must be the endpoint of the Web Service itself (i.e. the URL used to retrieve the WSDL document via the standard URL suffix, "?wsdl").

The Identify response must include an oai:description element containing a single Resource element with an xsi:type attribute that sets the element's type to vg:Registry. The content of vg:Registry type must be the registry description of the harvestable registry itself.

7.13 Harvesters

A registry that collects resource descriptions from other registries through the Harvesting Interface is referred to as a harvester. A full registry attempts to establish a complete collection of all resource descriptions known to the VO either by replicating the contents of another full registry, or—more commonly—by selectively harvesting from all known publishing registries. Typically in the latter case, the harvester periodically engages the ListRecords operation of each know publishing registry with the metadataPrefix parameter set to "ivo_vor", the set parameter set to "ivo_managed", and the from parameter set to the time of the last successful harvest for that publishing registry.

Any registry that claims to be a full registry must accept all records it harvests that are compliant with the VOResource metadata standard, even if the resource type is not one that is recognized by the registry.

7.14 Registering Registries

This specification defines a VOResource extension schema called VORegistry that can be used to specifically describe a registry and its support for the registry interface described in this document. These descriptions can be stored as resource records in registries. The schema is also used to register a naming authority—a publisher who claims ownership of an authority identifier from which IVOA identifiers may be created. A publishing registry is said to exclusively manage a

naming authority on behalf of the owning publisher; this means that only that registry may publish records with IVOA identifiers using that authority identifier.

The VORegistry schema namespace is "http://www.ivoa.net/xml/VORegistry/v1.0". As with the core VOResource Schema, the namespace URI has been chosen to allow it to be resolved as a URL to the XML Schema document that defines the VORegistry schema. Applications may assume that the namespace URI is so resolvable. In particular, it is recommended the namespace URI be given as the location for the VORegistry schema within the xsi:schemaLocation attribute."

Before a publisher can create resource descriptions using a new authority identifier, it must first register its claim to the authority identifier by creating a vg:Authority record.

As a subclass of vr:Service, the vg:Registry type uses vr:capability elements to describe its support for the interfaces described in this specification.

A registry declares itself to be a searchable registry by including a vr:capability element with an xsi:type attribute set to vg:Search.

A registry declares itself to be a searchable registry by including a vr:capability element with an xsi:type attribute set to vg:Search.