VOTable: A Proposed XML Format for Astronomical Tables

Alberto **Accomazzi**, Harvard-Smithsonian Center for Astrophysics, USA
Daniel **Durand**, Dominion Astrophysical Observatory, Canada
Pierre **Fernique**, Observatoire Astronomique de Strasbourg, France
Robert **Hanisch**, Space Telescope Science Institute, USA
Tom **McGlynn**, NASA Goddard Space Flight Center, USA
John **Good**, NASA Infrared Processing and Analysis Center, USA
Andrew **Lawrence**, Royal Observatory Edinburgh, UK
François **Ochsenbein**, Observatoire Astronomique de Strasbourg, France
Peter **Quinn**, European Southern Observatory, Germany
Ed **Shaya**, NASA Goddard Space Flight Center, USA
Alex **Szalay**, Johns Hopkins University, USA
Marc **Wenger**, Observatoire Astronomique de Strasbourg, France
Andreas **Wicenec**, European Southern Observatory, Germany
Roy **Williams**, California Institute of Technology, USA

1. Introduction

The VOTable format is a proposed XML standard for representing a table. In this context, a table is an unordered set of records, each of a uniform format. Each record is a sequence of primitive data types, together with metadata about the meaning of the data. The format is derived from the Astrores format [1], and backward compatible with that standard, except for (a) Fields are no longer allowed outside a Table, and (b) the Format attribute – used for automatic parsing of sexagesimal input – is no longer supported. Astrores was modeled on the FITS Binary Table format [2].

1.1. Example

A simple example of a VOTable document is:

```
<?xml version="1.0"?>
<!DOCTYPE ASTRO SYSTEM "http://. . ./VOTable.dtd">
<ASTRO ID="v1.0">
<RESOURCE>
<TABLE>
  <NAME>Stars</NAME>
  <DESCRIPTION>Some bright stars</DESCRIPTION>

  <FIELD ID="Star Name" ucd="ID_MAIN"
    datatype="A" width="10"></FIELD>

  <FIELD ID="RA" ucd="POS_EQ_RA"
    unit="degrees" datatype="E" precision="5"></FIELD>

  <FIELD ID="Dec" ucd="POS_EQ_DEC"
    unit="degrees" datatype="E" precision="5"></FIELD>

  <DATA><CSV colsep="|"><STREAM><![CDATA[
Procyon|114.827242|5.227506
Vega|279.234106|38.782992
]]></STREAM></CSV></DATA>

</TABLE>
</RESOURCE>
</ASTRO>
```

This table shows the positions of two stars, each with a name and two floating point numbers as coordinates. The star names have a fixed length of 10 characters, (shorter names will be padded by trailing blanks). The floating-point numbers (RA and Dec) are in degrees, and assumed to have five significant

digits (`precision="5"`), irrespective of the number of digits presented in the data. The frame of the coordinate system is not specified, although it can be with the `COOSYS` element.

1.2. XML

XML (extensible Markup Language) is powerful standard now throughout the Internet industries. It derives through simplification from SGML, which has been a standard in 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 contain `attributes` (keyword-value combinations), such as the `FIELD` elements above.

1.3. Syntax policy

The element names are in uppercase in order to help the reading. The attribute names are preferably in lowercase (with an exception for the `ID` attribute). Element and attribute names are further distinguished in this paper by being in `Lucida Console fixed-width font`.

1.4.   Remarks about the ID attribute

VOTable uses the ID attribute defined by Xpointer standard in order to refer to other elements in the document. The attribute `ID` can take any authorized values in XML, and each ID *must be unique* in the XML document. For example `ref="apple"` refers to the element that contains `ID="apple"` in the current XML document. Elements that may have ID tags are `ASTRO`, `COOSYS`, `FIELD`, `INFO`, `LINK`, `RESOURCE`, `TABLE`, and `VALUES`. Elements that support the `ref` attribute (and can point to those with ID) are: `CELL`, `FIELD`, and `TABLE`.

The ID is different from the `name` attribute in that (a) the ID attribute must be unique (or else the document is considered invalid in the XML sense), whereas names need not be unique; and (b) There should be support in the parsing software to look up references and extract the relevant element with matching ID. It should be noted that this referencing mechanism will not work unless the parser uses a *validating* parser.

2. Semantics of a VOTable

In this section we define the semantics of a VOTable, and in the next sections its syntax. A table has two sections, metadata and data – see figure. The metadata describes the table itself (name, title, description, and an optional coordinate system), and the nature of each field (column) of the table is defined by the `FIELD` element. There may also be `STREAM` objects that are intended to connect either the table or its records to external data sources through local files, ftp, http, gridftp, or other protocols. The address of the remote object is written in the URL syntax, `protocol://resource:port/file`.

A Table in this context is illustrated below. The top line of the table is a class definition (metadata) for all the instances (also known as rows, or records) of data in the subsequent lines. The VOTable document may contain the data part of the table, or it may not. If it does not contain data, there may be a pointer to the data; this would be best if the data is large, as XML tools may become unreliable for very large data sets. Each row of the table is a set of instances of primitive types, such as float, int, doubleComplex, and so on – see table below for complete list. There may also be strings and blobs for holding binary content. These may have the same length in each row, or each instance may have a different length. The semantic meaning of a blob (eg."This is a JPEG image") is not defined by VOTable, but it may be written into the description or name fields, or the ID mechanism discussed above.

Each `FIELD` (or column) of the table is defined by the nature of the primitive data, and by name, description, units, and info attributes. There is also a Universal Content Descriptor (UCD), which is a reference into a glossary created at CDS Strasbourg. Another attribute is the precision, which expresses the implied accuracy (number of significant digits) of each datum in this column.

To finish this section on semantics, we should note that a VOTable document may be used to express a question as well as an answer. Suppose there is a table that has no data – it has all the metadata (header)

Table **Stars** (Some bright stars)

StarName (10-char string) is a: ID_MAIN	RA (degrees) (4-byte float, 5 s.f.) is a: POS_EQ_RA	Dec (degrees) (4-byte float, 5 s.f.) is a: POS_EQ_DEC
Procyon	114.827242	5.227506
Vega	279.234106	38.782992

fields, as above, but no actual data rows. Then we could think of this document as a form that is to be filled in, as a request for data; the specification of *class* as an implicit request for *instance*. This distinction is made concrete in the type attribute of the TABLE element:

- results and data mean the same thing, that this is an instance of a table, and it should have data.

- meta and query and form mean the same, and imply that the table is to be viewed as a request for data rather than a statement of the data.

3. Metadata Content

The Table is written in XML as a NAME, TITLE, DESCRIPTION, LINK fields, that describe the nature of the data in the table. The LINK field may be parsed (see section 3.4). There is may be a COOSYS element, that contains specific information on the astronomical coordinate system that is being used. The rest of the metadata describes the FIELDs that together make up each row of the table.

A field has several attributes, including the informational NAME, TITLE, DESCRIPTION, and LINK, as well as VALUES, that can express limits and ranges of the values that the corresponding cell can contain, such as minimum, maximum, or enumeration of possible values. As explained in the previous section, it may also use the type attribute to differentiate a query for data from the data itself.

The FIELD must contain a datatype attribute, that expresses the nature of the data that is in the cells of this column of the table. This determines how data is read and stored internally. If it is not present, an exception is thrown.

datatype value	Type	Length (bytes)
L	Logical	1
I	Short Integer	2
J	Integer	4
K	Long integer	8
E	Floating point	4
D, F	Double	8
C	Float Complex	8
M	Double Complex	16
A	ASCII Character String	-
B	Binary Blob	-
U	Unicode String	-
X	Bit field	-

Some data types always have fixed size, such as Integer (J) and Double (D). For others (A, B, U, X), the size may be the same for every table cell in that column (*uniform* size), and the width attribute of the FIELD element can be used to specify this. The width of the bit datatype represents the number of 8-bit bytes that are used, and the width of the String (A) and

Binary (B) primitives may also be fixed in the metadata. Strings and Blobs shorter than this setting are right-padded with null characters (a zero byte), and longer ones are truncated to the uniform size.

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 on ASCII strings (`datatype="A"`), but also Unicode (`datatype="U"`).

The String and Blob primitives may also have a size that varies from row to row of the table, so that, for example, a JPEG image could be associated with each row of the table. It should be pointed out that the storage and processing of uniform-length strings and blobs will be much more efficient that that of variable-length.

For details of the exact meaning of these data types, please see section 8.

3.1. Numerical Accuracy

The VOTable format is meant for transferring, storing, and processing tabular data, it is not intended for presentation purposes. Therefore we carefully avoid giving rules on presentation, such as formatting. However, for non-string datatypes the `width` attribute of the `FIELD` is meant to suggest the number of characters to use for input or output of the quantity.

But there is a semantic difference between a number written as "5.12" and one that is written "5.1200", in that the former implies three significant digits of accuracy, and the latter five digits. Therefore the number of digits to show is not purely a presentation matter, but part of the metadata content of the number.

VOTable provides the `precision` attribute in the `FIELD` element to express the number of significant digits, or equivalently, the log of the implied error estimate of the numbers in the column. More control is available through an initial character: setting this to "E" rather than the default "F" implies that the `precision` measures is relative error (significant figures) rather than absolute error (decimal places). Thus `precision="E5"` means an implied relative error 10^{-5}, and `precision="5"` or "F5" means an implied absolute error 10^{-5}.

3.2. Universal Content Descriptors

The CDS in Strasbourg has used the metadata from thousands of astronomical tables to make a hierarchical glossary of the scientific meanings of the data in those tables [3]. Of 1600 entries in the glossary, here are a few typical examples.

`PHOT_INT-MAG_B`	Integrated total blue magnitude
`ORBIT_ECCENTRICITY`	Orbital eccentricity
`STAT_MEDIAN`	Statistics Median Value
`INST_QE`	Detector's Quantum Efficiency

The `ucd` attribute of the `FIELD` is to hold this information.

3.3. VALUES element

The `VALUES` element of the `FIELD` is designed to hold subsidiary information about the nature of the data in the field. It may have `MIN` and `MAX` elements, and it may contain `OPTION` elements. The latter contains name and value attributes, and may also contain more `OPTION` elements, so that a hierarchy of keyword-values pairs may be associated with each field.

There may also be a `null` attribute. If this is present, and a table cell takes this value, it is assumed to mean that no data is present. For example, there may be a convention that missing values in a table are expressed with –99, in which case the `NULL` element would be set to this. Therefore any cell in this field with this value is assumed to have no data.

There may also be an attribute called "`invalid`", meaning that this value should be used in case a table cell cannot be read. If, for example a row of a table should be all integers, and it's CSV representation is:

```
34, 45, 11, ---, 76
```

In this case, the unparsable value will cause an exception to be thrown, unless the relevant field definition contained something like:

```
<VALUES invalid="-99">
```

in which case the cell with the bad text would contain the integer –99 instead.

3.4. LINK Fields as URL Templates

The LINK element is to provide pointers to other documents or data servers on the Internet through a URL. In Astrores, the LINK element may be part of the RESOURCE, TABLE or FIELD elements. The href attribute of the LINK is meant to provide a URL that is at least valid syntactically, even though there need be no assurance that the link will actually connect and deliver data. It may be that a strange protocol is implied that the parser does not know about, for example gridftp://server/file. However, parsers are expected to understand at least the file, http and ftp protocols.

The gref attribute is meant for a higher-level protocol of some type, perhaps a logical name for a data resource, perhaps a GLU reference [4].

In some cases, there is additional semantics for the LINK element, where the href and gref attributes are not a simple URL, but rather a template for creating URL's. Depending on the content-role attribute of the LINK, and the nature of the parent element, the ID tags from the table may be substituted into the template to create an implicit new column, as explained in the next section.

3.4.1. Pattern-matching and Substitution

When a LINK element appears within a TABLE, there is extra functionality implied. The href or gref attributes may not be a simple link, but instead a template for a link. For example, in the table of section 1.1, we might have:

```
<LINK href="http://us-vo.org/lookup?Star=${Star Name}&RA=${RA}&DE=${Dec}"/>
```

The implication is that the text is seen in the context of a particular row of the table, and a substitution filter is applied. If the selected row of the table is the first one, the result of the substitution would be:

```
http://us-vo.org/lookup?Star=Procyon&RA=114.827&DE=5.227
```

Whenever the pattern ${…} is found in the original link, the part in the braces is compared with the set of name attributes of the fields of the table. If a match is found, then the value from that field of the selected row is used in place of the ${…}. If no match is found, no substitution is made. Thus the parser makes available to the calling application a value of the href and gref attributes that depends on which row of the table has been selected. Another way to think of it is that there is not a single link associated with the table, but rather an implicitly defined new column of the table. This mechanism can be used to connect each row of the table to further information resources.

The action attribute in this release of the standard is simply a string. In a future release, it may gain an implied string substitution filter as with href and gref.

The purpose of the link is defined by the content-role attribute. The allowed values are query, hints, and doc. The first implies that string substitution should be used as defined above, and the latter two imply first that no substitution is needed, and that the link points to either information for use by the application (hints) or human-readable documentation (doc).

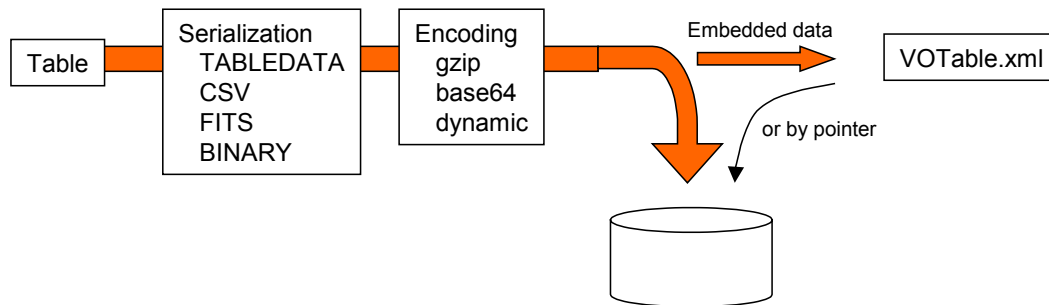## 3.5. Type Attribute

The type attribute of the FIELD may carry values that express the status of this field when the enclosing table is a query, rather than a data document. If the value is hidden, then a form asking for values will not ask for values of the hidden field. If the value is noquery, then %%%%. If the value is trigger, then %%%%.

4. Data Content

While the bulk of the metadata of a VOTable document is in the FIELD elements, the data content of the table is in a single DATA element. The data is organized in "reading" order, so that the content of each row appears in the same order as the order of the FIELD tags, with each row having the same number of items as there are FIELD tags.

The figure below shows how the abstract table is rendered into the VOTable document. First the data is *serialized*, as XML or CSV (column separated values, or a FITS binary table, or a simple binary format). This data stream may then be *encoded*, perhaps for compression or to convert binary to text. Finally, the data stream may be put in a remote file with a URL-type pointer in the VOTable document; or the table data may be embedded in the VOTable.

The data section of the VOTable document is created through a data pipeline. The abstract table is first serialized by one of several methods, then Encoded for compression or other reasons. It may be embedded in the xml file (*local* data), or it may be *remote* data.

4.1. Data Serialization

The serialization elements and their attributes are:

4.1.1.   TABLEDATA

This element is a way to build the table in pure XML, and is the only serialization method that does not allow an encoding or a remote data stream. It contains ROW elements, which in turn contain CELL elements. An example:

```
<TABLEDATA>
<ROW> <CELL>Procyon</CELL> <CELL>114.827242</CELL> <CELL>5.227506</CELL>  </ROW>
<ROW> <CELL>Vega</CELL>    <CELL>279.234106</CELL> <CELL>38.782992</CELL> </ROW>
</TABLEDATA>
```

While this serialization has a high overhead in the number of bytes, it has the advantage that XML tools can manipulate and present the table data directly.

Each item in the CELL tag is passed to a reader that is implicitly defined by the datatype attribute of the corresponding FIELD, which attempts to read the object from it. If it reads a value that is the same as the NULL value for that field, then the cell is assumed to contain no data.

The reader may not succeed, for example if we try to parse the string 245.6h756 into a string, then we succeed, but we cannot parse it into a float. In this case, the value from the attribute named "invalid" is used from the field, if present, or a NaN is used for the floating point FIELDs, or an exception is thrown for non-floating-point fields.

The TABLEDATA element may have an attribute to define the handling of complex numbers. If a CELL contains a complex number, it should be encoded as two numbers with a separator character between them. This character may be defined by the complexsep attribute:

```
<TABLEDATA complexsep=",">
```

4.1.2. CSV

The CSV (Column Separated Values) element contains the table data as a single XML element, with a specified *separator* character (`colsep` attribute) between the items of a row, a specified row separator (`rowsep` attribute) that separates rows of a table, and a number of headlines (`headlines` attribute), that are ignored by the reading software. For example:

```
<CSV colsep="|" rowsep="#" headlines="2">
Table of stars
These first two lines are ignored
Procyon | 114.827242 | 5.227506 # Vega |
279.234106 | 38.782992 #
</CSV>
```

In order to parse this, we first remove the headlines, which are ended by newline characters ('\n' in C). The text is now tokenized according to the row separator character (by default it is newline). Each row of the table is tokenized by the column separator character (by default it is comma). Each of the resulting strings is trimmed, meaning that consecutive whitespace characters are removed from the beginning or end of each token string. Each resulting string is then used to read a number in the same way as the previous section.

The CSV element may also use the `complexsep` attribute, as in the previous section, to define how complex numbers are serialized.

4.1.3. FITS

The FITS format for binary tables is well-used in astronomy [2], and its structure is a major influence on the VOTable specification. Metadata is stored in a header section, followed by the data. The metadata is substantially equivalent to the metadata of the VOTable format. One important difference is that VOTable does not require specification of the number of rows in the table, an important freedom is the table is being created dynamically from a stream.

The VOTable specification does not define the behavior of of parsers with respect to this doubling of the metadata. A parser may ignore the FITS metadata, or it may compare it with the VOTable metadata for consistency, or other possibilities.

4.1.4. BINARY

The Binary format is intended to be easy to read by parsers, so that additional libraries are not required. It is just a sequence of byte strings, the length of each string corresponding to the `datatype` attributes of the FIELD elements in the metadata. More details are in Appendix B, which is modified from the FITS specification document.

Strings (`datatype="A"` and "U") and blobs (`datatype="B"`) may have uniform or variable length. In the former case, the number of bytes is the same for each instance of the item, as specified by the `width` attribute of the FIELD. In the case of variable-length strings and blobs, however, the Binary format becomes more complex, and a second data stream is needed (see below).

4.2. Data Encoding

As a result of the serialization, the table has been converted to a byte stream, either text or binary. If the TABLEDATA serialization is used, then those elements are directly in the XML document, and there is no possibility for encoding. However, if one of the other serializations is used, we might *encode* the result to compress it, or for other reasons.

In this version of VOTable, it is not possible to encode individual columns of the table: The whole table must be encoded in the same way.

In order to use an encoding of the data, it must be enclosed in a STREAM element, whose attributes define the nature of the encoding. The `encoding` attribute is a string that should indicate to the parser how to undo the encoding that has been applied. Parsers should understand and interpret these values at least:

- encoding="gzip" implies that the data following has been compressed with the gzip filter, so that gunzip or similar should be applied.

- encoding="base64" implies that the base64 filter has been applied, to convert binary to text.

The parser may also respond to the string dynamic, implying that the data is in a remote resource (see below), and the encoding will be delivered with the header of the data. This occurs with the http protocol, where the MIME header indicates the type of encoding that has been used.

The default value of the encoding attribute is the null string, meaning that no encoding has been applied.

4.3. Remote Data

If the encoding of the data produces text, or if the serialization is naturally text-based, then it can be directly embedded into the XML document. However, if the data encoding produces binary, or if the data is very large, it may be preferable to keep the data separate from the metadata. In this case, there is STREAM with the attribute remote="yes".

The text contained in the STREAM element is then interpreted as the location of the data, rather than the data itself. The location is specified in a URL-type syntax, for example:

```
<STREAM remote="yes">ftp://server.com/mydata.dat</STREAM>
<STREAM remote="yes">http://webserver.com/mydata.dat</STREAM>
<STREAM remote="yes">gridftp://server.com/mydata.dat</STREAM>
<STREAM remote="yes">file://mydata.dat</STREAM>
```

The examples are the well-known anonymous ftp, and http protocols. Also is an example of a Grid-based access to data, and finally a local file, which is on the local file system.

There are two attributes of the STREAM element that may be useful. The expires tag is for when the VOTable is part of a pipeline of data processing, when data is being dynamically created and stored in temporary space, in which case it may be deleted after a certain time limit. The expires attribute expresses when a remote resource may cease to become valid, and is expressed in the same way as in the HTTP specification [4], for example:

```
  <STREAM expires="Wed, 26 Feb 1997 08:21:57 GMT">
```

The rights attribute expresses authentication information that may be necessary to access the remote resource. If the VOTable document is suitable encrypted, this attribute could be used to store a password.

5. Strings and Blobs

While some columns of the table contain fixed-length data types such as 4-byte integers or 8-byte doubles, the string (datatype="A" and "U") and blob (datatype="B") types may have uniform length or variable length. If there is a width attribute, that provides the number of bytes that every value has for the whole column. If there is no width attribute, of if the width is not positive, it is assumed that the string or blob is of variable width, with each value in the column having a different number of bytes.

Variable-length blobs can be used to store, for example a JPEG image in each row of the table, where the number of bytes depends on the complexity of the image. However, if all strings/blobs of the table are of uniform length, it implies greater efficiency in accessing and processing the data stream.

Variable length interaction with Binary Serialization

Just as the FITS binary table deals with variable-length objects, so does the Binary format. The data is split into a table of uniform-length items and a "heap" table of the variable-length items. The variable-length columns contain pointers into the heap.

The FITS file is organized with the uniform table first, and the heap of variable length items following. To create this file, it is necessary to know the number of rows in the table before starting the output of the file. With the Binary format, however, we have chosen to separate the two pieces to enable streaming of

VOTable documents between processors. If the data is serialized as BINARY, and there are variable-length FIELDs, then there should be two streams, as in the following example.

```
<TABLE>
  <FIELD name="RA" datatype="E"/>
  <FIELD name="Dec" datatype="E"/>
  <FIELD name="image" datatype="B" width="0">
  <DATA><BINARY>
    <STREAM remote="yes" type="data">file://table.dat</STREAM>
    <STREAM remote="yes" type="heap">file://heap.dat</STREAM>
  </BINARY></DATA>
</TABLE>
```

6. Document Structure

The VOTable document consists of a single all-containing element called ASTRO, which may contain a DESCRIPTION and a number of INFO elements which contain strings, a DEFINITIONS element, and a RESOURCE element.

6.1. DEFINITIONS element

This element may contain a definition of a coordinate system, stored in a COOSYS element, that is a *system*, (eg. Eq_FK5, galactic), and an equinox and epoch. A custom coordinate system may be specified as the "xy" value of the system, possibly with the attributes "x" and "y" for additional information. There may also be a USER_SPACE element that may contain user-specific data. Each of these may have an ID attribute, that can be referenced with the ref attribute of a field. Thus we can achieve grouping of fields (by having members of the group reference the same part of the definitions sections. We can also extend the definition of a field by adding user-specific data.

6.2. RESOURCE element

There may be multiple RESOURCE elements, and each of these may contain a NAME. TITLE, INFO and DESCRIPTION elements, as well as DEFINITIONS (as above). There may be LINK elements to provide URL-type pointers that give further information.

The main ingredient of the RESOURCE element is one or more TABLEs. These are described in sections 3-5 of this document.

7. Differences from Astrores

Data formats

Addition of two binary formats for the data section:

- FITS Binary Table, and

- Binary format, which is essentially the FITS format but without the header, except that while the FITS specification requires a keyword for the number of records in the table, the BINARY format does not.

Remote Data

In VOTable the data part of the table may be in a different file, on a server on another machine, or at the end of some other sort of socket. However the XML metadata document expresses the meaning and syntax of the data sufficiently well to read and understand the data when necessary. The data element may contain a STREAM elements; if the attribute remote="yes", then a URL syntax expresses the location. Note that the syntax file:// can be used for local files.

Encoded data

VOTable has the possibility that the data part of the document has been *encoded*. The encoding attribute of a STREAM is expressed by an arbitrary string, but parsers should correctly interpret "gzip" and "base64".

The encoding may also be written as `"dynamic"`, in which case the parser should discover the encoding dynamically as it reads the headers of the data resource, for example the http headers.

However an exception is that data of type TABLEDATA (full XML tagging) may not be encoded.

Data expiration and Rights Management

In VOTable, the remote STREAM element used to point to remote data should allow two further attributes:

- `expires` implies that the data under the link will not be available after a given time, as with the HTTP specification.

- `rights` is a string that will hold information about who is allowed to access the data under the link.

`datatype`, `precision`, and `width`

The variety of datatypes that may appear in tables is expanded, including 64-bit integers and complex types. The `precision` attribute is used to express the nature of the implied error in a quantity. The `width` attribute is used in two distinct ways: for strings and blobs to express that they have uniform size for every row of the table; and for other datatypes as a hint to a presentation system about how much horizontal space to use.

User Specific Information

The DEFINITIONS section of the VOTable document already contains document-wide information such as the coordinate system that is being used. In VOTable, it also contain a "user specific" element where other such information can be placed. In particular, it may be referenced from a table attribute to add semantic information such as the meaning of a blob field (eg. "this is a jpeg image"). Another use might be to group fields together, perhaps to state that two numbers for a sky position should be grouped as a single entity.

Format Attribute Removed

We suggest that the `format` attribute of the field element be removed from the specification. This has been used for example as `format="%RAh %RAm %RAs"` so that sexagesimal values can be read from the table. We feel that the effort of fully defining and implementing the formatting language is greater than the utility of the attribute.

Variable Length Strings and Blobs

If the `width` attribute is not specified, or not a positive integer, for a string or blob, then it is assumed to be variable length. The Binary data format must then have a new and separate section called HEAP, where these items are stored, and a pointer into this heap is kept in the table itself. This mechanism follows the FITS specification for variable length arrays.

Version and Namespace

The root element of the document (ASTRO) now may have a version attribute, anticipating future version control of VOTable documents. Furthermore, the tags used in the VOTable document should be part of an explicit XML Namespace, so that they can be used as part of a larger document, for example a report or web page. Without the namespace specification, there could be a collision of names (eg. TABLE). This change involves only one line of the table.

8. Adapted from FITS Binary Table Specification

Logical If the value of the `datatype` attribute specifies data type L, the contents of the field shall consist of ASCII T indicating true or ASCII F, indicating false. A 0 byte (hexadecimal 0) indicates an invalid value.

Bit Array If the value of the `datatype` attribute specifies data type X, the contents of the field shall consist of a sequence of bits starting with the most significant bit; the bits following shall be in order of decreasing significance, ending with the least significant bit. A bit array shall be composed of an integral number of bytes, with those bits following the end of the data set to zero.

Character If the value of the `datatype` attribute specifies data type A, the field shall contain a character string of zero or more members, composed of ASCII text. This character string may be terminated before

the length specified by the repeat count by an ASCII NULL (hexadecimal code 00). Characters after the first ASCII NULL are not defined. A string with the number of characters specified by the repeat count is not NULL terminated. Null strings are defined by the presence of an ASCII NULL as the first character.

Unicode Character If the value of the `datatype` attribute specifies data type `U`, the field shall contain a character string of zero or more members, composed of Unicode text. Each character is represented by two bytes, in order that many non-Latin alphabets can be represented.

Unsigned 8-Bit Integer If the value of the `datatype` attribute specifies data type `B`, the data in the field shall consist of an array of unsigned 8-bit integers. This array of bytes is also known as a "blob", and can be used for storing general byte data.

16-Bit Integer If the value of the `datatype` attribute specifies data type `I`, the data in the field shall consist of twos-complement signed 16-bit integers, contained in two bytes. The most significant byte shall be first. Within each byte the most significant bit shall be first, and subsequent bits shall be in order of decreasing significance. An exception is thrown for invalid values, unless the `invalid` attribute has been set, in which case it is used instead.

32-Bit Integer If the value of the `datatype` attribute specifies data type `J`, the data in the field shall consist of twos-complement signed 32-bit integers, contained in four bytes. The most significant byte shall be first, and subsequent bytes shall be in order of decreasing significance. Within each byte, the most significant bit shall be first, and subsequent bits shall be in order of decreasing significance. An exception is thrown for invalid values, unless the `invalid` attribute has been set, in which case it is used instead.

Single Precision Floating Point If the value of the `datatype` attribute specifies data type `E`, the data in field shall consist of ANSI/IEEE-754 [15] 32-bit floating point numbers. All IEEE special values are recognized. The IEEE NaN is used to represent invalid values, unless the `invalid` attribute has been set, in which case it is used instead.

Double Precision Floating Point If the value of the `datatype` attribute specifies data type `D` or `F`, the data in the field shall consist of ANSI/IEEE-754 64-bit double precision floating point numbers. All IEEE special values are recognized. The IEEE NaN is used to represent invalid values, unless the `invalid` attribute has been set, in which case it is used instead.

Single Precision Complex If the value of the `datatype` attribute specifies data type `C`, the data in the field shall consist of a sequence of pairs of 32-bit single precision floating point numbers. The first member of each pair shall represent the real part of a complex number, and the second member shall represent the imaginary part of that complex number. If either member contains a NaN, the entire complex value is invalid.

Double Precision Complex If the value of the `datatype` attribute specifies data type `M`, the data in the field shall consist of a sequence of pairs of 64-bit double precision floating point numbers. The first member of each pair shall represent the real part of a complex number, and the second member of the pair shall represent the imaginary part of that complex number. If either member contains a NaN, the entire complex value is invalid.

9. Sample VOTable Document

```
<!DOCTYPE ASTRO SYSTEM "VOTable.dtd">
<ASTRO version="v0.9" xmlns="http://vizier.u-strasbg.fr/VOTable">

    <DESCRIPTION about=http://astrores.xml.org/ASTRO-Spec-0.2 />

    <DEFINITIONS>
        <COOSYS ID="myJ2000" system="eq_FK5" equinox="2000." epoch="2000."/>
    </DEFINITIONS>

    <!-- The output is made of several tables related together -->
    <RESOURCE ID="I254">
        <NAME>GSC1.2</NAME>
        <TITLE>The HST Guide Star Catalog, Version 1.2 (Lasker+ 1996)</TITLE>
        <DESCRIPTION>
    This is an excerpt of the GSC1.2.  This version was re-reduced with PPM catalogue; see more details about the GSC
catalogues at http://www-gsss.stsci.edu/gsc/gsc.html .
        </DESCRIPTION>

        <TABLE ID="gsc_out">
            <NAME>gsc.out</NAME>
            <TITLE>Output from GSC1.2 Server</TITLE>
            <DESCRIPTION>    Default result of GSC1.2 Server around a target</DESCRIPTION>

            <FIELD ID="_r" ucd="POS_ANG_DIST" unit="arcmin" datatype="F" width="7" precision="4">
                <NAME>_r</NAME>
                <DESCRIPTION>Distance from target NGC40</DESCRIPTION>
                <VALUES type="actual">
                    <MIN value="0.0"/>
                    <MAX value="10.0"/>
                </VALUES>
            </FIELD>

            <FIELD ID="gsc" datatype="A" width="10" ucd="IDENT">
                <NAME>GSC-Id</NAME>
                <TITLE>Unique object id</TITLE>
                <DESCRIPTION>The GSC-Id is made of 10 digits, 5 representing the place number, and 5 the object
number on the plate. </DESCRIPTION>
            </FIELD>

            <FIELD ID="ra" ref="myJ2000" ucd="POS_EQ_RA" unit="deg" datatype="F" precision="5">
                <NAME>RA(J2000)</NAME>
                <DESCRIPTION>Right ascension in J2000, epoch of plate</DESCRIPTION>
            </FIELD>

            <FIELD ID="dec" ref="myJ2000" ucd="POS_EQ_DE" unit="deg" datatype="F" precision="5">
                <NAME>Dec(J2000)</NAME>
                <DESCRIPTION>Declination in J2000, epoch of plate</DESCRIPTION>
            </FIELD>

            <FIELD ID="pos_err" unit="arcsec" datatype="F" precision="1" ucd="ERROR">
                <NAME>PosErr</NAME>
                <DESCRIPTION>Mean error on position</DESCRIPTION>
            </FIELD>

            <FIELD ID="Pmag" ucd="PHOT_PHG_MAG" unit="mag" datatype="F" precision="2">
                <NAME>Pmag</NAME>
                <DESCRIPTION>photographic magnitude (see n_Pmag)</DESCRIPTION>
            </FIELD>

            <FIELD ID="e_Pmag" ucd="ERROR" unit="mag" datatype="F" precision="2">
                <NAME>e_Pmag</NAME>
                <DESCRIPTION>Mean error on photographic magnitude</DESCRIPTION>
            </FIELD>
```

```xml
<FIELD ID="class" ucd="CLASS_CODE" datatype="I">
    <NAME>Class</NAME>
    <DESCRIPTION>Class of object (0=star; 3=non-stellar)</DESCRIPTION>
    <VALUES type="actual">
        <OPTION name="star" value="0"/>
        <OPTION name="galaxy" value="3"/>
    </VALUES>
</FIELD>
<LINK content-role="doc" title="documentation" href="http://vizier.u-strasbg.fr/viz-bin/Cat?I/254"/>

<DATA>
    <TABLEDATA>
<ROW><CELL>0.0146</CELL><CELL>0430201297</CELL><CELL>4.7766</CELL><CELL>72.8474</CELL><CELL>3.6</CELL><CELL>8.59 </CELL><CELL>0.20</CELL><CELL>0</CELL></ROW>
<ROW><CELL>0.9704</CELL><CELL>0430200545</CELL><CELL>5.4576</CELL><CELL>72.6528</CELL><CELL>0.2</CELL><CELL>12.18</CELL><CELL>0.34</CELL><CELL>0</CELL></ROW>
<ROW><CELL>0.9730</CELL><CELL>0430200545</CELL><CELL>3.9867</CELL><CELL>72.9484</CELL><CELL>0.2</CELL><CELL>12.09</CELL><CELL>0.20</CELL><CELL>0</CELL></ROW>
<ROW><CELL>1.5843</CELL><CELL>0430202363</CELL><CELL>8.9587</CELL><CELL>72.6635</CELL><CELL>0.2</CELL><CELL>14.38</CELL><CELL>0.34</CELL><CELL>0</CELL></ROW>
<ROW><CELL>2.8586</CELL><CELL>0430200269</CELL><CELL>5.4847</CELL><CELL>72.8272</CELL><CELL>0.3</CELL><CELL>14.96</CELL><CELL>0.20</CELL><CELL>3</CELL></ROW>
<ROW><CELL>2.9198</CELL><CELL>0430200153</CELL><CELL>10.4746</CELL><CELL>72.4542</CELL><CELL>0.2</CELL><CELL>12.89</CELL><CELL>0.20</CELL><CELL>0</CELL></ROW>
<ROW><CELL>2.9215</CELL><CELL>0430200153</CELL><CELL>6.9484</CELL><CELL>72.1162</CELL><CELL>0.2</CELL><CELL>13.06</CELL><CELL>0.34</CELL><CELL>0</CELL></ROW>
<ROW><CELL>3.0487</CELL><CELL>0430202336</CELL><CELL>4.7586</CELL><CELL>72.9837</CELL><CELL>0.2</CELL><CELL>14.38</CELL><CELL>0.34</CELL><CELL>0</CELL></ROW>
<ROW><CELL>3.2247</CELL><CELL>0430200121</CELL><CELL>7.9585</CELL><CELL>72.5565</CELL><CELL>0.2</CELL><CELL>12.39</CELL><CELL>0.21</CELL><CELL>0</CELL></ROW>
<ROW><CELL>3.2269</CELL><CELL>0430200121</CELL><CELL>7.9484</CELL><CELL>72.5874</CELL><CELL>0.2</CELL><CELL>12.50</CELL><CELL>0.34</CELL><CELL>0</CELL></ROW>
    </TABLEDATA>
</DATA>
</TABLE>
</RESOURCE>
</ASTRO>
```

10. The DTD for VOTable

```
<!ELEMENT ASTRO (DESCRIPTION?, DEFINITIONS?, INFO*, RESOURCE+)>
<!ATTLIST ASTRO
    ID ID #IMPLIED
    version CDATA #IMPLIED
>
<!ELEMENT DESCRIPTION (#PCDATA)>
<!ATTLIST DESCRIPTION
    about CDATA #IMPLIED
>
<!ELEMENT DEFINITIONS (USER_SPACE?, COOSYS*)>
<!ELEMENT INFO (#PCDATA)>
<!ATTLIST INFO
    ID ID #IMPLIED
    name CDATA #IMPLIED
    value CDATA #IMPLIED
>
<!ELEMENT RESOURCE (DEFINITIONS*, NAME?, TITLE?, DESCRIPTION?, INFO*, TABLE*, LINK*)>
<!ATTLIST RESOURCE
    ID ID #IMPLIED
>
<!ELEMENT NAME (#PCDATA)>
<!ELEMENT TITLE (#PCDATA)>
<!ELEMENT TABLE (NAME?, TITLE?, DESCRIPTION?, COOSYS*, FIELD*, LINK*, DATA?)>
<!ATTLIST TABLE
    ID ID #IMPLIED
    ref IDREF #IMPLIED
    type (results|data    |    meta|query|form) "results"
>
<!ELEMENT FIELD (NAME?, TITLE?, DESCRIPTION?, VALUES*, LINK?)>
<!ATTLIST FIELD
    ID ID #IMPLIED
    unit CDATA #IMPLIED
    datatype (L | X | B | I | J | A | F | D | C | M | K) #IMPLIED
    precision CDATA #IMPLIED
    width CDATA #IMPLIED
    ref IDREF #IMPLIED
    name CDATA #IMPLIED
    ucd CDATA #IMPLIED
    type (hidden | no_query | trigger) #IMPLIED
>
<!ELEMENT VALUES (MIN*, MAX*, OPTION*)>
<!ATTLIST VALUES
    ID ID #IMPLIED
    multiple (yes | no) "no"
    type (legal | actual) "legal"
    null CDATA #IMPLIED
    invalid CDATA #IMPLIED
>
<!ELEMENT MIN (#PCDATA)>
<!ATTLIST MIN
    value CDATA #REQUIRED
    inclusive (yes | no) "yes"
>
<!ELEMENT MAX (#PCDATA)>
<!ATTLIST MAX
    value CDATA #REQUIRED
    inclusive (yes | no) "yes"
>
<!ELEMENT OPTION (OPTION*)>
<!ATTLIST OPTION
    name CDATA #IMPLIED
    value CDATA #REQUIRED
>
<!ELEMENT LINK (#PCDATA)>
<!ATTLIST LINK
    ID ID #IMPLIED
    content-role (query | hints | doc) #IMPLIED
```

```
        content-type CDATA #IMPLIED
        title CDATA #IMPLIED
        value CDATA #IMPLIED
        href CDATA #IMPLIED
        gref CDATA #IMPLIED
        action CDATA #IMPLIED
>
<!ELEMENT STREAM (#PCDATA)>
<!ATTLIST STREAM
        remote (yes | no) "no"
        type (data | heap) "data"
        encoding CDATA #IMPLIED
        expires CDATA #IMPLIED
        rights CDATA #IMPLIED
>
<!ELEMENT DATA (TABLEDATA | BINARY | FITS | CSV)>
<!ELEMENT TABLEDATA (ROW*)>
<!ELEMENT ROW (CELL+)>
<!ELEMENT CELL (#PCDATA)>
<!ATTLIST CELL
        ref IDREF #IMPLIED
>
<!ELEMENT FITS (#PCDATA | STREAM)*>
<!ELEMENT CSV (#PCDATA | STREAM)*>
<!ATTLIST CSV
        recsep CDATA #IMPLIED
        colsep CDATA #IMPLIED
        headlines CDATA #IMPLIED
        complexsep CDATA #IMPLIED
>
<!ELEMENT BINARY (#PCDATA | STREAM)*>

<!ELEMENT USER_SPACE (#PCDATA | VALUES)*>
<!ATTLIST USER_SPACE
        ID ID #IMPLIED
>
<!ELEMENT COOSYS (#PCDATA)>
<!ATTLIST COOSYS
        ID ID #IMPLIED
        system (eq_FK4 | eq_FK5 | ICRS | ecl_FK4 | ecl_FK5 | galactic | supergalactic | xy | barycentric | geo_app) "eq_FK5"
        equinox CDATA #IMPLIED
        epoch CDATA #IMPLIED
        x CDATA #IMPLIED
        y CDATA #IMPLIED
>
```

11. References

[1] Accomazzi *et. al, Describing Astronomical Catalogues and Query Results with XML*
http://vizier.u-strasbg.fr/doc/astrores.htx

[2] *FITS: Flexible Image Transport Specification*, specifically the Binary Tables Extension
http://fits.gsfc.nasa.gov/

[3] *Universal Content Descriptors*
http://vizier.u-strasbg.fr/doc/UCD.htx

[4] *GLU: Générateur de Liens Uniformes, CDS Strasbourg*
http://simbad.u-strasbg.fr/glu/glu.htx

[5] *World Wide Web Consortium, HTTP 1.1 Specification*
http://www.w3.org/Protocols/rfc2068/rfc2068