

Unicode in VOTable

VOTable towards v1.6?

Mark Taylor (Bristol)

IVOA Interop
Görlitz

Applications WG

15 November 2025

`$Id: votable.tex,v 1.21 2025/11/13 14:04:22 mbt Exp $`

Outline

- Unicode in VOTable
 - Proposal
 - Consequences
 - Actions
- Other things for v1.6
- DALI xtype="strings"
- Roadmap/Discussion

Proposal for Unicode in VOTable

Problems with current VOTable ($\leq v1.5$):

- `datatype="char"` restricted to 7-bit ASCII
- `datatype="unicodeChar"` restricted to *Basic Multilingual Plane*^{*}, also inefficient **BINARY** serialization
→ No emoji possible in VOTable 😞
- VOTable discusses Unicode in outdated terms (using the obsolete “UCS-2” encoding)
 - ▷ <https://wiki.ivoa.net/internal/IVOA/InterOpJune2025Apps/unicode-notes.pdf>
 - ▷ <https://wiki.ivoa.net/internal/IVOA/InterOpOct2014Applications/vot-unicode.pdf>

Proposal: VOTable **PR #71**

- `datatype="char"`:
 - ▷ 7-bit ASCII characters → UTF-8 bytes
- `datatype="unicodeChar"`:
 - ▷ UCS-2 characters → BMP-only UTF-16 byte pairs (this is just a change of terminology)
 - ▷ also **deprecated** in favour of `char`
- `arraysize` corresponds to count of **code units** not **characters**:
 - ▷ `char`: UTF-8 code unit = 1 octet
 - ▷ `unicodeChar`: UTF-16 code unit = 2 octets

} Required to make
BINARY/BINARY2
serializations work

^{*} *BMP: 65 536 code points covering almost all modern languages and symbols, but not emojis and some weird stuff*

Unicode: Consequences 1

Intended consequences of proposal at VOTable 1.6:

- Any Unicode character can be written in a `datatype="char"` column
 - ▷ `TABLEDATA`: use document encoding
 - ▷ `BINARY/BINARY2`: use UTF-8 encoding
- BMP characters *can* be written in a `datatype="unicodeChar"` column
 - ▷ `TABLEDATA`: use document encoding
 - ▷ `BINARY/BINARY2`: use UTF-16 encoding
 - ▷ ... but don't do it, because it's now **deprecated**

Unicode: Consequences 2

Unintended corollaries at VOTable 1.6:

- You can't specify a string column with a fixed number of characters
 - ▷ you have to specify the length of the UTF-8/UTF-16 **serialization** instead
 - ▷ ... unless e.g. you know the column is 7-bit ASCII
- Single (scalar) `datatype="char"` columns can still only contain 7-bit ASCII
 - ▷ ... since non-ASCII code points need multiple bytes in UTF-8
- String truncation is not straightforward
 - ▷ Overlength strings may need to be truncated to fit in fixed-`arraysize` strings/array elements
 - ▷ Such truncation has to be done carefully (not in the middle of a multi-octet UTF-8 character)
- Decoding string arrays (multi-dimensional `char/unicodeChar` arrays) requires unpacking to bytes then counting code units (i.e. counting bytes) **not** counting characters
 - ▷ This may be a bit surprising to implementors, but it's not so hard

Unicode: Backward Compatibility

New code reading existing legal VOTables

- Will read correctly

Old code reading new VOTables

- Will mostly read them as intended
- But some problems possible with non-ASCII multi-dimensional char arrays (string arrays)

```
<VOTABLE version="1.6" xmlns="http://www.ivoa.net/xml/VOTable/v1.3">
<RESOURCE>
<TABLE>
<FIELD name="places" datatype="char" arraysize="10x4"/>
<DATA>
<TABLEDATA>
  <TR><TD>0123456789012345678901234567890123456789</TD></TR>
  <TR><TD>Valletta..Coll. ParkGörlitz..Strasbourg</TD></TR>
</TABLEDATA>
</DATA>
</TABLE>
</RESOURCE>
</VOTABLE>
```

“Görlitz..” is 10 *UTF-8 code units* (bytes)
but 9 *Unicode code points* (characters)

New reader:

```
+-----+
| places                                     |
+-----+
| (0123456789, 0123456789, 0123456789, 0123456789) |
| (Valletta.., Coll. Park, Görlitz.., Strasbourg) |
+-----+
```

Old reader:

```
+-----+
| places                                     |
+-----+
| (0123456789, 0123456789, 0123456789, 0123456789) |
| (Valletta.., Coll. Park, G?rlitz..S, trasbourg) |
+-----+
```

Unicode: Implications for Implementations

Changes required by implementors of VOTable I/O libraries

- Input:

- ▷ Code for decoding multi-dimensional character arrays (i.e. 1+-dimensional string arrays) will need changing
 - Unpacking array elements now requires counting bytes (code points) not characters
- ▷ Other input code: read Unicode/UTF-8 — maybe no changes?
 - If you're not trying too hard in a Unicode-aware language, there's a good chance that VOTable-reading code is doing the right thing already (reading `TABLEDATA` in document encoding, `char BINARY/2` as UTF-8)
 - If you're doing careful validation or sanitisation of VOTable input you will need to change code (mostly: relax checks, remove special cases)
 - If you're manipulating VOTables in FORTRAN 77 you may have some work to do
- ▷ No special casing required for older VOTables
 - Unicode-friendly implementation is also correct for existing VOTable versions

- Output:

- ▷ Stop writing `FIELDS/PARAMs` with `datatype="unicodeChar"` (now deprecated and unnecessary)
- ▷ Remove checks for unwriteable characters in character output?
- ▷ Write character data using Unicode encoding (may have been doing that already)
- ▷ Careful writing fixed-length (including scalar) character array, i.e. string-valued, `FIELDS/PARAMs`
 - For 1-d character arrays (scalar strings): easiest to just always use variable-length strings (`arraysize="*"`)
 - To support string arrays (can't have variable-length strings) or fixed-length strings: byte counting not character counting now required; care needed with string truncation

Unicode: PARAM and INFO

In all VOTable versions ≤ 1.5 :

- `datatype="char"` must be 7-bit ASCII
- `datatype="unicodeChar"` must be BMP-only

... everywhere, including

- `BINARY/BINARY2`
- `TABLEDATA`
- `PARAM`
- and even `INFO`:

VOTable 1.5 sec 4.8: *“The INFO element is a PARAM element restricted to be of type string (i.e. `datatype="char"` and `arraysize="*"` are implied).”*

- Consequences:
 - ▷ My prototype VOTable 1.6-compatible STIL now handles these “correctly”
 - `votlint` VOTable validator reports an error
 - STIL maps non-ASCII `char` bytes/non-BMP `unicodeChar` pairs to `'?'`
 - ▷ Should it do that? There’s no reason for the restriction in `PARAM/INFO` (since binary encoding is never required)
 - ▷ Should we have an **Erratum** for `PARAM/INFO`?

Unicode: PARAM and INFO

In all VOTable versions ≤ 1.5 :

- `datatype="char"` must be 7-bit ASCII
- `datatype="unicodeChar"` must be BMP-only

... everywhere, including

- `BINARY/BINARY2`
- `TABLEDATA`
- `PARAM`
- and even `INFO`:

VOTable 1.5 sec 4.8: *“The INFO element is a PARAM element restricted to be of type string (i.e. `datatype="char"` and `arraysize="*"` are implied).”*

- Consequences:
 - ▷ My prototype VOTable 1.6-compatible STIL now handles these “correctly”
 - `votlint` VOTable validator reports an error
 - STIL maps non-ASCII `char` bytes/non-BMP `unicodeChar` pairs to `'?'`
 - ▷ Should it do that? There’s no reason for the restriction in `PARAM/INFO` (since binary encoding is never required)
 - ▷ Should we have an **Erratum** for `PARAM/INFO`?

This is illegal:

```
<INFO name="location" value="Görlitz"/>
```

Unicode: Status

PR #71 Redefine char and unicodeChar for correct Unicode usage

- Discussion so far:
 - ▷ Approved (following some edits) by Russ Allbery and Gregory D-F (Rubin)
 - ▷ Discussed at Astro-CC meeting Trieste Oct 2025 (Markus, Mark T et al.)
 - ▷ Approved (with some suggestions) by Tom Donaldson
 - ▷ Comments from the audience?
- Implementations:
 - ▷ Implemented and working in STIL ($\rightarrow$ STILTS, TOPCAT); [prototype](#) available, tests ongoing
 - ▷ Will need ≥ 1 more — Astropy?
- Suggested next step:
 - ▷ Post summary of proposed changes + pointer to PR to Apps list soon/now
 - ▷ If no objections, merge PR soon — end Nov 2025?

Unicode content in INFO/PARAM

- This hasn't had much discussion (I only realised it this month)
- Suggest to draft an Erratum to apply to **all** VOTable versions ≤ 1.5 :
INFO and char/unicodeChar PARAM may contain unrestricted Unicode content

Other Items

Other uncontroversial(?) things for VOTable 1.6

- Define `content` parameter for VOTable MIME type ([#26](#), [#15](#))
 - ▷ Datalink recommends `application/x-votable+xml; content=datalink`, but `content` parameter is currently undefined
- Review Appendix A “*Possible VOTable extensions*” ([#53](#))
 - ▷ Appendix A, untouched since v1.10 (2004), has several suggestions of unclear status; remove or extract to Note?
- Some editorial issues
- ... any more?

Probably not for VOTable 1.6:

- Some suggestions without clear consensus [#72](#), [#29](#)
- Some issues tagged v2.0 [#23](#), [#19](#), [#17](#)
- See also [#25](#) → DALI issue [#66](#) `xtype="strings"`

DAI xtype="strings"

String arrays in VOTable are a pain.

- VOTable datatype `char` is a single character; there is no fundamental string datatype (*following FITS*)
 - ▷ Encode a string as a 1-d character array
 - may be fixed-length string (`arraysize="8"`)
or variable-length string (`arraysize="*"`)
 - ▷ Encode a 1-d array of strings as a (rectangular) 2-d character array
 - may be fixed-length array of fixed-length strings (`arraysize="8x3"`)
or variable-length array of fixed-length strings (`arraysize="8x*"`)
 - may **not** be array of **variable-length strings** (only last dimension can vary)
- But array of variable-length strings is often what you want
 - ▷ Otherwise you have to count all string lengths up front
- Also: assembling string arrays into rectangular char arrays is fiddly
 - ▷ especially with the UTF-8 encoding rule proposed for VOTable 1.6

```
<FIELD name="colours"
      datatype="char"
      arraysize="6x3"/>

...
<TD>Red___PurpleBlue__</TD>
<TD>Green_Cyan__Black_</TD>
```

DALI xtype="strings"

String arrays in VOTable are a pain.

- VOTable datatype `char` is a single character; there is no fundamental string datatype (*following FITS*)
 - ▷ Encode a string as a 1-d character array
 - may be fixed-length string (`arraysize="8"`)
or variable-length string (`arraysize="*"`)
 - ▷ Encode a 1-d array of strings as a (rectangular) 2-d character array
 - may be fixed-length array of fixed-length strings (`arraysize="8x3"`)
or variable-length array of fixed-length strings (`arraysize="8x*"`)
 - may **not** be array of **variable-length strings** (only last dimension can vary)
- But array of variable-length strings is often what you want
 - ▷ Otherwise you have to count all string lengths up front
- Also: assembling string arrays into rectangular char arrays is fiddly
 - ▷ especially with the UTF-8 encoding rule proposed for VOTable 1.6

```
<FIELD name="colours"
      datatype="char"
      arraysize="6x3"/>
...
<TD>Red___PurpleBlue__</TD>
<TD>Green_Cyan__Black_</TD>
```

Alternative: encode 1-d string arrays using delimiters

- See [DALI issue #66](#)
- Still experimental
- Currently: delimiter "|" with some escaping mechanism
- Basic VOTable reader reads string, aware VOTable reader reads array
- Doesn't help with higher-dimensional string arrays ... but they are rare

```
<FIELD name="colours"
      datatype="char"
      arraysize="*" xtype="strings"/>
...
<TD>Red|Purple|Blue</TD>
<TD>Green|Cyan|Black</TD>
```

Roadmap

Suggested next steps:

- Start preparing VOTable 1.6
 - ▷ Invite final mailing list comments on [PR #71](#) (Unicode)
 - ▷ Prepare PRs for issues [#26](#) (MIME type content parameter), [#53](#) (review Appendix A)
 - ▷ Address editorial issues
 - ▷ VOTable 1.6 WD by next Interop?
 - ▷ Lead editor: Mark T (discussed with Tom D)
- Prepare Erratum on Unicode [PARAM/INFO](#) content?
- Prototype/experiment with `xtype="strings"` → DALI-next?

Comments?