Unicode in VOTable

Markus Demleitner

College Park IVOA Interop, June 2025

Do We Have a Problem?

You: "There's the datatype unicodeChar, so what's wrong?"

Me: "Not muuuuuch. But it still sucks."

Problem I: UCS-2

VOTable 1.1 through 1.5 say:

> Each Unicode character is represented in the BINARY/BINARY2 serialization by two bytes, using the big-endian UCS-2 encoding (ISO-10646-UCS-2).

But:

```
$ python3 -c 'b"abc".decode("ucs-2")'
Traceback (most recent call last):
  File "<string>", line 1, in <module>
LookupError: unknown encoding: ucs-2
```

Even worse: UCS-2 cannot encode Emojis!!!!!!!!!

What's With UCS-2?

UCS-2 was the original two-bytes-per-codepoint encoding that was the Unicode group's favourite until they noticed 2^{16} glyphs aren't enough for Emojis.

With Unicode 2.0 (1996), they replaced it with UTF-16. UTF-16 uses reserved code points in the BMP ("surrogates") to encode out-of-BMP code points.

In that sense, UTF-16 is backwards-compatible with UCS-2. And UCS-2 has been deprecated forever.

Minimal Measure

We ought to change the VOTable spec to replace all references to UCS-2 with references to UTF-16.

But…

The following VOTable is invalid:

```
<VOTABLE><RESOURCE><TABLE>
<FIELD name="objname" datatype="char" arraysize="*"/>
<DATA><TABLEDATA><TR>
<TD>Joachim Wambsganß</TD>
</TR></TABLEDATA></DATA></TABLE></RESOURCE></VOTABLE>
```

Can you spot the problem?

Problem II: 7-Bit Chars

When VOTable was devised, there were still lots of 8 bit-encodings around. For instance, `\xC4` could mean Ä, an uppercase Delta, a cryillic EF, or an arabic Waw with a Hazma in ISO 8859 alone.

It was wise to not commit to any concrete encoding and to say: "stay within ASCII and you're safe".

These days are over.

UTF-8 is everywhere, and that's unlikely to change before we achieve superluminal space travel.

Deprecate unicodeChar?

On the other hand, WHATWG says: "UTF-16BE/LE, which [is] *unfortunately* required due to deployed content" (emph. mine).

At least they would like to get rid of UTF-16.

And so would, frankly, I. It would also fix the above VOTable in one fell swoop.

What if we said "char may have UTF-8 in it, and we'll drop unicodeChar a few decades from now"?

Trouble?

I admit:

- `<FIELD datatype="char"/>` still can only keep ASCII (as *any* non-ASCII needs a second byte in encoded form).
- `<FIELD datatype="char" arraysize="<n>"/>` doesn't mean "there's n unicode codepoints in the string". It means "The utf-8 representation of this is n bytes long."

Is either of that a problem?

I'd say: It's just a wart smaller than any alternative we have.

Except…

Well… *if* we could find it in ourselves to define a proper, variable-length string datatype in VOTable, that would still be better and we could deprecate both char and unicodeChar arrays as stand-ins for strings.

I'd be in on that effort, too.