

Re: Data table text I/O package?

Source: <http://coding.derkeiler.com/Archive/Ada/comp.lang.ada/2005-06/msg00295.html>

- *From:* Georg Bauhaus <bauhaus@xxxxxxxxxxxxxx>
 - *Date:* Tue, 21 Jun 2005 23:01:59 +0200
-

Dmitry A. Kazakov wrote:

Let me first guess that many here have their largely regular and homogenous data in mind. I'm not talking about this. We went off from what to do if you don't have atomic, homogenous, unambiguous data, sent around.

- 1) If you have a nice arrangement of exactly one set of array-like data of guaranteed quality, there is little to win by using XML.
- 2) Given a data format much like in (1), if you can pick up the phone and ring the other end of the data-sending connection, and say, 'Uhm, we have seen a slight change in the data text table, could you explain ...' or similar, you are privileged.
- 3) I you think that every bunch of data is sent in agreeable format, I could be telling you a few stories, though not in public.

Back to step one: brackets in computer tables are not named, a computer doesn't have accountants' abilities in pattern matching when looking at rows and columns in a table. Again, I said XML is good for parsing of data if you cannot tell in advance that the data stream is totally free of errors.

Re: Data table text I/O package?

No it is bad, because missing one bracket may lead to loss of the whole data set. As a medium XML is as awful as readable.

If you mean loosing a closing tag, the parser can correct, though not always, and to different extents. If you mean somehow a '>' of a start tag is lost, then this is better or worse than in typical CSV or similar; a line end is a bracket, too. A separator is a two-way bracket, adding one more possibility for error and ambiguity.

Imagine a CSV stream with `_no_` record separators. (This is not fiction.) It is kind of efficient, you count fields.

However, if some data item contains a separator due to an error, you loose the whole stream, or use the wrong data without noticing this, in the worst case.

>>As for whitespace, read Stroustrup's article on defining operator
whitespace.

Delimiter /= whitespace.

True, still Stroustrup demonstrates some effects we are discussing.

You're missing the point: XML is **not** about rendering data.

Sorry, but the thread's subject reads "Data table text I/O package". Text = rendered data.

Notice that the thread title has I/O. I/O can mean pretty printing, and it can mean a reliable and robust data input-output facility, working well in the face of erroneous input.

I was under the impression that we were discussing the latter, in particular I added: You better had such-and-such data if you want to reliably handle data in a sloppy setting, answering

Re: Data table text I/O package?

Marius Amado Alves IIRC.

Logarithms are logarithms, not printed logarithms, this is a second step. Data formats for exchange or storage on the one hand and a print-out of some data on the other hand are two very different beasts, with different purposes. Consider the MVC paradigm.

This is obviously wrong, clearly print-outs serve both data exchange and data storage when humans are involved.

The point is whether print-outs serve **well** as a data exchange format, IN THE SITUATION described above, that is you do not know in advance that you will get the finest data. I doubt that this is the case in any but a few well defined situations. (I.e., you might meet it more often in contexts where Ada is used, or so I hope.)

The accuracy is well defined and most importantly, it is up to the application, yours and mine respectively.

This is a wrong approach of course.

There is no more accurate representation of 3.15 than the text "3.15", right under our noses. In a text data stream, tabular, XML, whatever.

I appreciate that you care how I should read a "3.15" and store it. Though, if my application uses decimal fixed point to represent money with 4 digits after the point, then you can add as many zeros as you like after .15, it's none of your business, it's the other application's business. I may not have your hardware, I may not have your rounding policies. I still find your data useful.

Because the accuracy of the data is **not** defined by the internal type used.

The accuracy of the data may not be defined at all, IN THE DATA

Re: Data table text I/O package?

STREAM. (The again, some peoply may try, adding a schema.)

3.15 is as accurate as can be, and independent of bits.

Is it 3.14998751 or 3.150000?

It is 3.15. This is data, text data. Not a computer floating point value, just data in textual external format, very flexible, and with the number of digits that you see. Do with it what your application wants to do with it. This is what you get. "Is the light On, or Off?" -- "It is On." Data, "Off" or "On". No matter how any program represents On or Off, all that can be said about On or Off AS PARTS OF THE DATA IN THE TEXT STREAM is in the stream. Use a Boolean, or use an enumeration type, your choice.

A data stream does not in general define semantics. On the contrary, the standards talk about applications defining meaning, in the end.)

Floating-point numbers are intervals.
Transporting them you should either use explicit bounds:

Who said floating point? I said "3.15", ('3', '.', '1', '5'). You do not have have a solution to the problem of exactly representing R-eal values in a data transport context, do you? (And note that not every important number originates inside a computer's FPU.)

[3.1499, 3.1600]

Well, someone will ask you, 'and what exactly is 3.1499?' on *our* machine?

Really? A normal log file [...]

You argue from your log files, let me argue from a heterogeneity point of view. (BTW, I use text pipes and stream analysis to look at files of about this size.)

Re: Data table text I/O package?

A server is running, you can look at the trace log, some parser fails, you want to know why. Say there are three lines of ';'-separated data, each at most 400 characters long. Ideally one appears right after the other. These lines are what they send you, no way to change that. Each field has varying length. Your job will be to associate matching fields. Because 370 characters don't fit in a single display line, you end up counting ';'s in each line and take notes, or c&p, to find the matching fields.

Now consider separated key=value lines. They will be longer, but you can scan the line looking for the key strings. A big step up. XML isn't worse in my view.

That is: the names, the types and units are *factored* out to the table header, which allows the reader to concentrate on the *values*. Thus a table looks as:

Distance [km]	Temperature [°C]	...
3.15	29.0	...
2.10	14.4	...

This is readable.

Sure, I'm of course not saying a table isn't readable. I can even use XSL-FO or TeX to produce a table from XML, no problem. In fact, I have done this many times.

A formatted table just isn't that robust. Consider the case where the headline gets lost. The missing redundancy will leave you with a puzzle, not a robust set of self describing text.

This is irrelevant in data exchange. This is print.

To make difference more visible, consider bitmaps stored XML format. Would you be able to recognize a person's face in it?

Re: Data table text I/O package?

I'm sure you know one can make text images, but I won't argue about this for the same reasons that have been explained for years when discussing SGML and data for which no parsing is desired.

So an image is not print whereas a table is?

Now we are entering the realm of robust image encoding...
No.

Georg Bauhaus