

Re: Did Borland doing well in Q4? Listen to the Earning CC

Source:

<http://coding.derkeiler.com/Archive/Delphi/borland.public.delphi.non-technical/2008-03/msg00155.html>

- *From:* Jolyon Smith <jsmith@xxxxxxxxxxxxxx>
 - *Date:* Tue, 4 Mar 2008 08:39:34 +1300
-

In article <xn0fn8gnrld5rjx00rnewsgroups@xxxxxxxxxxxxxx>, newsgroups@xxxxxxxxxxxxxx says...

Jolyon Smith wrote:

So to ensure that your ANSI application continues to behave correctly AS an ANSI application you will need to change all "String" declarations to "ANSIString" (and Char to ANSICChar and PChar to ANSICChar) etc.

Only THEN can you start thinking about how Unicode is best implemented and supported in your application and begin the process of migrating to Unicode.

Sounds dramatic. I doubt it will be, in reality. <shrug>

Funny, you seemed to think I didn't know of what I spoke, yet it is transparently clear that you don't have a clue what problems many are going to have to deal with, and I have it on good authority that our specific problems are FAR from unique.

Anybody using FastStrings for example (and I *know* we are not alone in that respect).

My ASM isn't even barely passable so I can't say for certain, but my guess is that many if not all FastStrings routines are not going to "play nice" with UTF16 strings.

Even the specifically "ANSI" FS routines (FastANSIReplace) do not use explicit ANSI declarations in their prototypes, so at the very least these will need to be updated.

(iirc hasn't PM stopped working on/supporting FastStrings? How many

Re: Did Borland doing well in Q4? Listen to the Earning CC

other 3rd party libraries/components that people rely on are in the same boat?)

But supposing we get ANSIStrng declarations in those prototypes so that their innards at least are safely cocooned in an ANSI world, safe from the UTF16 maelstrom outside.....

....what happens when a user (peksy users!) then enters some Unicode data that doesn't losslessly convert to ANSI when passed to these routines internally (one must presume there will be auto-conversion when calling a proc accepting ANSI strings but passing UniCode strings).

After all, they will now be using a Unicode application with all that that implies, and free to enter any Unicode string their heart desires, all without the app developer ever having to lift a finger to enable it. Or, more worryingly, to DEAL with it.

But when that happens (UnicodeString implicitly convert to ANSIStrng), is the compiler going to emit warnings where-ever such implicit conversions are taking place?

I've seen mention of warning when "WIDECHAR reduced to CHAR", but that was in relation to the use of "set of Char", not ANSIStrng and UnicodeString.

Will it be the same?

Bottom line:

Despite how much we might wish it were so, supporting Unicode is not simply a matter of "turning it on", *especially* in an existing application.

There is many a slip twixt cup and lip – the cup here is the user data entry, the lip is your internal string handling. The potential slips are numerous.

(PS – if you know for a fact that FastStrings is going to "just work" in Tiburon, then please share that insight. It would at least be one less thing for me to have to worry about. Only ONE less, mind you.)

.