Subject: Re: 16 bits wchar
Posted by mirek on Mon, 01 Oct 2007 12:28:20 GMT
View Forum Message <> Reply to Message

cbpporter wrote on Mon, 01 October 2007 07:24
I think that the first thing that must be done is extended PutUtf8 and GetUtf8 so that it reads correctly th values outside of BMP. This is not too difficult and I will try to implement and test this.

I guess fixing Utf8 routines to provide UTF16 surrogate support (for now) is a good idea.

Quote:
The only issue is how to handle ill-formated values. I came to the conclusion that read and write operation must recognize compliant encodings, but it also must process ill-formated characters and insert them into the stream. If the stream is set to strict, it will throw an exception. If not, it will still encode. I propose the Least Complex Encoding TM possibility. Non-atomic Unicode aware string manipulation functions will should not fail when encountering such characters, so after a read, process and write, these ill-formated values (which could be essential to other applications) will be preserved. In this scenario, only functions that display the string must be aware that some data is ill-formated.

Well, the basic requirement there is that converting UTF8 with invalid sequences to WString and back must result in the equal String. This feat is successfully achieved by UTF8EE.

Also, I do not think that any string manipulation routine everywhere ever should be aware about UTF-8 or UTF-16 encoding. It is much practical to covert to WString, process and (eventually) convert it back. I think that in the long run, it might be even faster.

Quote:
Next, there should be a method to Validate the string, and a way to convert strings containing ill-formated string to error-escaped strings and back, so we can use atomic string processing if needed. This conversion should be done explicitly, so no general performance overhead is introduced.

```
bool    CheckUtf8(const String& src);
```

You can add CheckUtf16

Anyway, seriously, I believe that the ultimate solution is to go with sizeof(wchar) = 4... The only trouble is converting this to UTF16 and back in Win32 everywhere... OTOH, good news is that after the system code is fixed, the transition does not pose too much backwards compatibility problems...

Mirek