Ticket #5033 (new Bugs)

Property Tree JSON Parser cannot handle utf-8 string correctly.

Opened 2 years ago
Last modified 5 months ago

Reported by:	Lorin Liu <liu.lorin@…>	Owned by:	cornedbee
Milestone:	To Be Determined	Component:	property_tree
Version:	Boost 1.45.0	Severity:	Problem
Keywords:		Cc:	

Description

Please refer to the following code fragment.

```
    // This is a json utf-8 string {"value": "天津"}
    const char json[] = {0x7B, 0x22, 0x76, 0x61, 0x6C, 0x75, 0x65, 0x22, 0x
                         0x22, 0xE5, 0xA4, 0xA9, 0xE6, 0xB4, 0xA5, 0x22, 0x

    boost::property_tree::ptree pt;
    boost::format fmter("%1% : %2% \n");
    std::stringstream strm;
    std::string value;

    strm << json;

    read_json(strm, pt);

    value = pt.get<std::string>("value");

    // Print the individual char one by one.
    // However the wrong result appears. All chars printed to console are 0
    // And the expected result should be the chars of 0xE5, 0xA4, 0xA9, 0xE
    BOOST_FOREACH(char c, value)
        std::cout << (fmter % (int)(unsigned char) c % c) << std::endl;
```

After my investigation, this might be a bug in boost/property_tree/detail/json_parser_read.hpp.

My patch for this issue is as follows.

```
--- json_parser_read.hpp.orig   2010-12-24 15:49:06.000000000 +0800
+++ json_parser_read.hpp        2011-01-02 10:26:37.000000000 +0800
@@ -145,7 +145,7 @@
             a_unicode(context &c): c(c) { }
             void operator()(unsigned long u) const
             {
-                u = (std::min)(u, static_cast<unsigned long>((std::numeric
+                // u = (std::min)(u, static_cast<unsigned long>((std::nume
                 c.string += Ch(u);
             }
         };
```

Attachments

- property.tree.read.UTF-8.patch (951 bytes) - added by *Ilya Bobyr <ilya.bobyr@…>* 5 months ago.
  Property Tree JSON reader fix for UTF-8 encoded string

Change History

Changed 18 months ago by Tommy comment:1

I can confirm this bug. As in the previous code snippets, a_unicode::operator() will be called with 0xE5. Because std::numeric_limits<char>::max is 127, so std::min(0xE5, 127)==127 will be append to the result string.

Hope this bug can be fixed in the next release.

Changed 15 months ago by rshhh <ryushiro.sugehara@…> comment:2 follow-up: ↓ 3

I think the approach taken in the patch is not correct.
Since a single byte of UTF-8 string could take a value larger than the maximum that ***signed char*** could take(0x7F), I think that certain characters may overflow a ***Ch*** (or just ***char***) object.
I'm guessing that the issue I just wrote is exactly the reason why the original code is taking std::min() approach, am I correct? So if my opinion is right, I think we should store the UTF-8 character in a ***unsigned char*** sequence...

Changed 15 months ago by Tommy comment:3 in reply to: ↑ 2

Replying to rshhh <ryushiro.sugehara@…>:

> I think the approach taken in the patch is not correct.
> Since a single byte of UTF-8 string could take a value larger than the maximum that ***signed char*** could take(0x7F), I think that certain characters may overflow a ***Ch*** (or just ***char***) object.
> I'm guessing that the issue I just wrote is exactly the reason why the original code is taking std::min() approach, am I correct? So if my opinion is right, I think we should store the UTF-8 character in a ***unsigned char*** sequence...

I think the problem is : a_unicode SHOULD handle a unicode char, which can't fit in a Ch (or just char). Should mbrtowc()/wcrtomb() be used to convert between unicode(wchar_t) and Ch?

Changed 10 months ago by Jan Ciger <jan.ciger@…> comment:4

Just got bitten by the same bug. What is the recommended fix for this?

Changed 5 months ago by Ilya Bobyr <ilya.bobyr@…>

- **attachment** *property.tree.read.UTF-8.patch* added

Property Tree JSON reader fix for UTF-8 encoded string

Changed 5 months ago by Ilya Bobyr <ilya.bobyr@…> comment:5

While it is true, that char can not handle whole Unicode, it can still handle values larger than 0x7F if you view it as an unsigned integer. There was a fix for JSON writer in version 1.45 that makes it unconditionally view character type as unsigned thus allowing it to save UTF-8 encoded strings even if char is signed. Here is a similar patch for the JSON reader. While it still has std::min() in there it uses maximum value for unsigned char when clamping a character value been read.

This way JSON writer and JSON reader are doing the same kind of transformation to the characters and UTF-8 encoded strings can go through a full save/load cycle.