Ruby trunk - Bug #7282

Invalid UTF-8 from emoji allowed through silently

11/06/2012 11:51 AM - headius (Charles Nutter)

Status:	Closed		
Priority:	Normal		
Assignee:	naruse (Yui NARUSE)		
Target version:	2.1.0		
ruby -v:		**Backport:**	

Description

On my system, where the default encoding is UTF-8, the following should not parse:

ruby-2.0.0 -e 'p "Hello, \x96 world!\"}'

But it does. And it is apparently marked as "ok" as far as code range goes, because encoding to UTF-8 does not catch the problem:

system ~/projects/jruby $ ruby-1.9.3 -e 'p "{\"sample\": \"Hello, \x96 world!\"}".encode("UTF-8")'
"{\"sample\": \"Hello, \x96 world!\"}"

system ~/projects/jruby $ ruby-2.0.0 -e 'p "{\"sample\": \"Hello, \x96 world!\"}".encode("UTF-8")'
"{\"sample\": \"Hello, \x96 world!\"}"

Nor does character-walking:

system ~/projects/jruby $ ruby-1.9.3 -e '"Hello, \x96 world!".each_char {|x| print x}'
Hello, ? world!
system ~/projects/jruby $ ruby-2.0.0 -e '"Hello, \x96 world!".each_char {|x| print x}'
Hello, ? world!

Nor does []:

system ~/projects/jruby $ ruby-1.9.3 -e 'p "Hello, \x96 world!"[7]'
"\x96"

system ~/projects/jruby $ ruby-1.9.3 -e 'p "Hello, \x96 world!"[8]'
" "

system ~/projects/jruby $ ruby-2.0.0 -e 'p "Hello, \x96 world!"[7]'
"\x96"

system ~/projects/jruby $ ruby-2.0.0 -e 'p "Hello, \x96 world!"[8]'
" "

But the malformed String does get caught by transcoding to UTF-16:

system ~/projects/jruby $ ruby-1.9.3 -e 'p "{\"sample\": \"Hello, \x96 world!\"}".encode("UTF-16")'
-e:1:in encode': "\x96" on UTF-8 (Encoding::InvalidByteSequenceError)
from -e:1:in'

system ~/projects/jruby $ ruby-2.0.0 -e 'p "{\"sample\": \"Hello, \x96 world!\"}".encode("UTF-16")'
-e:1:in encode': "\x96" on UTF-8 (Encoding::InvalidByteSequenceError)
from -e:1:in'

Or by doing a simple regexp match:

system ~/projects/jruby $ ruby-1.9.3 -e '"Hello, \x96 world!".match /.+/'
-e:1:in match': invalid byte sequence in UTF-8 (ArgumentError)
from -e:1:inmatch'
from -e:1:in `'

system ~/projects/jruby $ ruby-2.0.0 -e '"Hello, \x96 world!".match /.+/'
-e:1:in match': invalid byte sequence in UTF-8 (ArgumentError)

from -e:1:inmatch'
from -e:1:in `'

And of course I am ignoring the fact that it should never have parsed to begin with.

This kind of inconsistency in rejecting malformed UTF-8 does not inspire a lot of confidence.

JRuby allows it through the parser (this is a bug) but does fail in other places because the string is malformed.

History

#1 - 11/06/2012 02:58 PM - usa (Usaku NAKAMURA)

- Category set to M17N

- Status changed from Open to Assigned

- Assignee set to naruse (Yui NARUSE)

- Target version set to 2.0.0

#2 - 11/06/2012 03:23 PM - duerst (Martin Dürst)

Hello Charles,

On 2012/11/06 11:51, headius (Charles Nutter) wrote:

> Issue #7282 has been reported by headius (Charles Nutter).
>
> ---
>
> Bug #7282: Invalid UTF-8 from emoji allowed through silently
> https://bugs.ruby-lang.org/issues/7282
>
> Author: headius (Charles Nutter)
> Status: Open
> Priority: Normal
> Assignee:
> Category:
> Target version:
> ruby -v: 2.0.0
>
> On my system, where the default encoding is UTF-8, the following should not parse:
>
> ruby-2.0.0 -e 'p "Hello, \x96 world!\"}'

It doesn't. It should be
ruby-2.0.0 -e 'p "Hello, \x96 world!"}'
or
ruby-2.0.0 -e 'p "Hello, \x96 world!\"}"'
or
ruby-2.0.0 -e 'p "Hello, \x96 world!"'
or some such. But apart from that, you are right.

I'm no longer sure, but I think at some point, there was an argument to
allow \x in UTF-8 literals, and a reason to not check. But I can't
remember what, and if we can't remember, when we'd better make it check.

> But it does. And it is apparently marked as "ok" as far as code range goes, because encoding to UTF-8 does not catch the problem:
>
> system ~/projects/jruby $ ruby-2.0.0 -e 'p "{\"sample\": \"Hello, \x96 world!\"}".encode("UTF-8")'
> "{\"sample\": \"Hello, \x96 world!\"}"

Encoding to the encoding you're already in is a no-op. See also
https://bugs.ruby-lang.org/issues/6321.

> Nor does character-walking:
>
> system ~/projects/jruby $ ruby-2.0.0 -e '"Hello, \x96 world!".each_char {|x| print x}'
> Hello, ? world!
>
> Nor does []:
>
> system ~/projects/jruby $ ruby-2.0.0 -e 'p "Hello, \x96 world!"[7]'
> "\x96"

The underlying machinery is the same.

> But the malformed String does get caught by transcoding to UTF-16:
>
> system ~/projects/jruby $ ruby-2.0.0 -e 'p "{\"sample\": \"Hello, \x96 world!\"}".encode("UTF-16")'
> -e:1:in encode': "\x96" on UTF-8 (Encoding::InvalidByteSequenceError)
> from -e:1:in'

Yes, here you're actually transcoding, so this is checked.

> Or by doing a simple regexp match:
>
> system ~/projects/jruby $ ruby-2.0.0 -e '"Hello, \x96 world!".match /.+/'
> -e:1:in match': invalid byte sequence in UTF-8 (ArgumentError)
> from -e:1:inmatch'
> from -e:1:in `'

We'd need to dig in the code to figure out why it happens here.

> And of course I am ignoring the fact that it should never have parsed to begin with.
>
> This kind of inconsistency in rejecting malformed UTF-8 does not inspire a lot of confidence.
>
> JRuby allows it through the parser (this is a bug) but does fail in other places because the string is malformed.

Overall, the idea (I think) is to hit a balance between efficiency and
correctness. But checking at parsing time would probably be rather
efficient at avoiding errors.

Regards, Martin.

#3 - 11/07/2012 01:39 AM - headius (Charles Nutter)

duerst (Martin Dürst) wrote:

> > On my system, where the default encoding is UTF-8, the following should not parse:
> >
> > ruby-2.0.0 -e 'p "Hello, \x96 world!\"}'
>
> It doesn't. It should be
> ruby-2.0.0 -e 'p "Hello, \x96 world!"}'
> or
> ruby-2.0.0 -e 'p "Hello, \x96 world!\"}"'
> or
> ruby-2.0.0 -e 'p "Hello, \x96 world!"'
> or some such. But apart from that, you are right.

Yeah sorry...I guess I was rushed filing this issue. The last one is what I was going for.

> I'm no longer sure, but I think at some point, there was an argument to
> allow \x in UTF-8 literals, and a reason to not check. But I can't
> remember what, and if we can't remember, when we'd better make it check.

Yes, it seems like either this string should be forced to ASCII-8BIT, or else it shouldn't be allowed to parse in the first place. It definitely should not parse *and* be marked as valid UTF-8.

> > But it does. And it is apparently marked as "ok" as far as code range goes, because encoding to UTF-8 does not catch the problem:
> >
> > system ~/projects/jruby $ ruby-2.0.0 -e 'p "{\"sample\": \"Hello, \x96 world!\"}".encode("UTF-8")'
> > "{\"sample\": \"Hello, \x96 world!\"}"
>
> Encoding to the encoding you're already in is a no-op. See also
> https://bugs.ruby-lang.org/issues/6321.

Thank you. I suspected as much and will make changes to JRuby (and RubySpec if needed). JRuby was always doing the transcoding, so it blew up here attempting to walk UTF-8 characters.

> Nor does character-walking:
>
> ```
> system ~/projects/jruby $ ruby-2.0.0 -e '"Hello, \x96 world!".each_char {|x| print x}'
> Hello, ? world!
> ```
>
> Nor does []:
>
> ```
> system ~/projects/jruby $ ruby-2.0.0 -e 'p "Hello, \x96 world!"[7]'
> "\x96"
> ```
>
> The underlying machinery is the same.

Makes sense. JRuby also allows these cases through. Perhaps both cases should fail once they encounter a non-7bit, non-surrogate byte like \x96?

> ```
> system ~/projects/jruby $ ruby-2.0.0 -e '"Hello, \x96 world!".match /.+/'
> -e:1:in match': invalid byte sequence in UTF-8 (ArgumentError)
> from -e:1:inmatch'
> from -e:1:in `'
> ```
>
> We'd need to dig in the code to figure out why it happens here.

Well, at the very least it would have to be using the encoding subsystem for Oniguruma/Onigmo to walk characters; that logic almost certainly rejects \x96.

#4 - 02/18/2013 08:52 PM - naruse (Yui NARUSE)

- Tracker changed from Bug to Misc

headius (Charles Nutter) wrote:

> duerst (Martin Dürst) wrote:
>
> > Nor does character-walking:
> >
> > ```
> > system ~/projects/jruby $ ruby-2.0.0 -e '"Hello, \x96 world!".each_char {|x| print x}'
> > Hello, ? world!
> > ```
> >
> > Nor does []:
> >
> > ```
> > system ~/projects/jruby $ ruby-2.0.0 -e 'p "Hello, \x96 world!"[7]'
> > "\x96"
> > ```
> >
> > The underlying machinery is the same.
>
> Makes sense. JRuby also allows these cases through. Perhaps both cases should fail once they encounter a non-7bit, non-surrogate byte like \x96?

On string index access, Ruby doesn't raise error even if it is invalid byte sequence.

> > ```
> > system ~/projects/jruby $ ruby-2.0.0 -e '"Hello, \x96 world!".match /.+/'
> > -e:1:in match': invalid byte sequence in UTF-8 (ArgumentError)
> > from -e:1:inmatch'
> > from -e:1:in `'
> > ```
> >
> > We'd need to dig in the code to figure out why it happens here.
>
> Well, at the very least it would have to be using the encoding subsystem for Oniguruma/Onigmo to walk characters; that logic almost certainly rejects \x96.

On regexp match, Ruby raises error.

#5 - 02/24/2013 09:19 PM - ko1 (Koichi Sasada)

- Target version changed from 2.0.0 to 2.1.0

naruse-san, what is the status of this ticket?

#6 - 02/24/2013 09:24 PM - naruse (Yui NARUSE)

ko1 (Koichi Sasada) wrote:

> naruse-san, what is the status of this ticket?

I don't understand what is the current problem of this ticket.
If headius has some issue, could you summarize it?
Or nothing, close this.

#7 - 03/09/2013 05:11 PM - headius (Charles Nutter)

A couple quick tests seem to work ok in 2.0.0. If all my original cases from the report work properly (i.e. fail properly) then this one is fixed. I have not confirmed all scenarios yet.

#8 - 03/13/2013 11:16 AM - naruse (Yui NARUSE)

- Status changed from Assigned to Closed

#9 - 04/18/2013 05:59 AM - naruse (Yui NARUSE)

- Tracker changed from Misc to Bug