# Ruby trunk - Feature #11801

## rb_inspect shouldn't raise error even if calling inspect returns non compatible strings

12/10/2015 04:22 AM - naruse (Yui NARUSE)

| | |
|---|---|
| **Status:** | Closed |
| **Priority:** | Normal |
| **Assignee:** | naruse (Yui NARUSE) |
| **Target version:** | |

**Description**

Current rb_inspect raises Encoding::Compatibility error if obj.inspect returns a string
whose encoding is not compatible with resulted string's encoding.
But the behavior is not useful on inspecting an array or a hash containing such strings
because it prevents to dump them.

Following patch changes to escape such strings like String#inspect.

```
diff --git a/object.c b/object.c
index d339ff9..1f73cb4 100644
--- a/object.c
+++ b/object.c
@@ -465,6 +465,7 @@ rb_any_to_s(VALUE obj)
     return str;
 }

+VALUE rb_str_escape(VALUE str);
 /*
  * If the default external encoding is ASCII compatible, the encoding of
  * the inspected result must be compatible with it.
@@ -478,11 +479,11 @@ rb_inspect(VALUE obj)
     rb_encoding *ext = rb_default_external_encoding();
     if (!rb_enc_asciicompat(ext)) {
    if (!rb_enc_str_asciionly_p(str))
-       rb_raise(rb_eEncCompatError, "inspected result must be ASCII only if default external enco
ding is ASCII incompatible");
+       return rb_str_escape(str);
    return str;
     }
     if (rb_enc_get(str) != ext && !rb_enc_str_asciionly_p(str))
-   rb_raise(rb_eEncCompatError, "inspected result must be ASCII only or use the default external
encoding");
+   return rb_str_escape(str);
     return str;
 }

diff --git a/string.c b/string.c
index e6df91d..319c516 100644
--- a/string.c
+++ b/string.c
@@ -5265,6 +5265,70 @@ rb_str_buf_cat_escaped_char(VALUE result, unsigned int c, int unicode_p)
     return l;
 }

+VALUE
+rb_str_escape(VALUE str)
+{
+    int encidx = ENCODING_GET(str);
+    rb_encoding *enc = rb_enc_from_index(encidx);
+    const char *p = RSTRING_PTR(str);
+    const char *pend = RSTRING_END(str);
+    const char *prev = p;
+    char buf[CHAR_ESC_LEN + 1];
+    VALUE result = rb_str_buf_new(0);
+    int unicode_p = rb_enc_unicode_p(enc);
```

```
+    int asciicompat = rb_enc_asciicompat(enc);
+
+    while (p < pend) {
+   unsigned int c, cc;
+   int n = rb_enc_precise_mbclen(p, pend, enc);
+        if (!MBCLEN_CHARFOUND_P(n)) {
+       if (p > prev) str_buf_cat(result, prev, p - prev);
+            n = rb_enc_mbminlen(enc);
+            if (pend < p + n)
+                n = (int)(pend - p);
+            while (n--) {
+                snprintf(buf, CHAR_ESC_LEN, "\\x%02X", *p & 0377);
+                str_buf_cat(result, buf, strlen(buf));
+                prev = ++p;
+            }
+       continue;
+   }
+        n = MBCLEN_CHARFOUND_LEN(n);
+   c = rb_enc_mbc_to_codepoint(p, pend, enc);
+   p += n;
+   switch (c) {
+     case '\n': cc = 'n'; break;
+     case '\r': cc = 'r'; break;
+     case '\t': cc = 't'; break;
+     case '\f': cc = 'f'; break;
+     case '\013': cc = 'v'; break;
+     case '\010': cc = 'b'; break;
+     case '\007': cc = 'a'; break;
+     case 033: cc = 'e'; break;
+     default: cc = 0; break;
+   }
+   if (cc) {
+       if (p - n > prev) str_buf_cat(result, prev, p - n - prev);
+       buf[0] = '\\';
+       buf[1] = (char)cc;
+       str_buf_cat(result, buf, 2);
+       prev = p;
+   }
+   else if (asciicompat && rb_enc_isascii(c, enc) && ISPRINT(c)) {
+   }
+   else {
+       if (p - n > prev) str_buf_cat(result, prev, p - n - prev);
+       rb_str_buf_cat_escaped_char(result, c, unicode_p);
+       prev = p;
+   }
+    }
+    if (p > prev) str_buf_cat(result, prev, p - prev);
+    ENCODING_CODERANGE_SET(result, rb_usascii_encindex(), ENC_CODERANGE_7BIT);
+
+    OBJ_INFECT_RAW(result, str);
+    return result;
+}
+
 /*
  * call-seq:
 *   str.inspect   -> string
diff --git a/test/ruby/test_m17n.rb b/test/ruby/test_m17n.rb
index ca25f85..2e9e65b 100644
--- a/test/ruby/test_m17n.rb
+++ b/test/ruby/test_m17n.rb
@@ -278,7 +278,7 @@ def test_object_utf16_32_inspect
         o = Object.new
         [Encoding::UTF_16BE, Encoding::UTF_16LE, Encoding::UTF_32BE, Encoding::UTF_32LE].each do
|e|
           o.instance_eval "undef inspect;def inspect;'abc'.encode('#{e}');end"
-          assert_raise(Encoding::CompatibilityError) { [o].inspect }
+          assert_equal '[abc]', [o].inspect
```

```
        end
      ensure
        Encoding.default_internal = orig_int
@@ -302,13 +302,18 @@ def o.inspect
     def o.inspect
       "abc".encode(Encoding.default_external)
     end
-    assert_raise(Encoding::CompatibilityError) { [o].inspect }
+    assert_equal '[abc]', [o].inspect

     Encoding.default_external = Encoding::US_ASCII
     def o.inspect
       "\u3042"
     end
-    assert_raise(Encoding::CompatibilityError) { [o].inspect }
+    assert_equal '[\u3042]', [o].inspect
+
+    def o.inspect
+      "\x82\xa0".force_encoding(Encoding::Windows_31J)
+    end
+    assert_equal '[\x{82A0}]', [o].inspect
   ensure
     Encoding.default_internal = orig_int
     Encoding.default_external = orig_ext
diff --git a/core/array/shared/inspect.rb b/core/array/shared/inspect.rb
index dc30518..a169fd0 100644
--- a/core/array/shared/inspect.rb
+++ b/core/array/shared/inspect.rb
@@ -83,9 +83,9 @@ describe :array_inspect, shared: true do

     it "raises if inspected result is not default external encoding" do
       utf_16be = mock("utf_16be")
-      utf_16be.should_receive(:inspect).and_return("utf_16be".encode!(Encoding::UTF_16BE))
+      utf_16be.should_receive(:inspect).and_return(%<"utf_16be \u3042">.encode!(Encoding::UTF_16B
E))

-      lambda { [utf_16be].send(@method) }.should raise_error(Encoding::CompatibilityError)
+      [utf_16be].send(@method).should == '["utf_16be \u3042"]'
     end
   end
 end
```

### Associated revisions

**Revision e3ab670a - 12/10/2015 06:57 PM - naruse (Yui NARUSE)**

- object.c (rb_inspect): dump inspected result with rb_str_escape()
  instead of raising Encoding::CompatibilityError. [Feature #11801]
- string.c (rb_str_escape): added to dump given string like
  rb_str_inspect without quotes and always dump in US-ASCII
  like rb_str_dump.

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@53027 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

**Revision 53027 - 12/10/2015 06:57 PM - naruse (Yui NARUSE)**

- object.c (rb_inspect): dump inspected result with rb_str_escape()
  instead of raising Encoding::CompatibilityError. [Feature #11801]
- string.c (rb_str_escape): added to dump given string like
  rb_str_inspect without quotes and always dump in US-ASCII
  like rb_str_dump.

**Revision 53027 - 12/10/2015 06:57 PM - naruse (Yui NARUSE)**

- object.c (rb_inspect): dump inspected result with rb_str_escape() instead of raising Encoding::CompatibilityError. [Feature #11801]
- string.c (rb_str_escape): added to dump given string like rb_str_inspect without quotes and always dump in US-ASCII like rb_str_dump.

**Revision 53027 - 12/10/2015 06:57 PM - naruse (Yui NARUSE)**

- object.c (rb_inspect): dump inspected result with rb_str_escape() instead of raising Encoding::CompatibilityError. [Feature #11801]
- string.c (rb_str_escape): added to dump given string like rb_str_inspect without quotes and always dump in US-ASCII like rb_str_dump.

**Revision 53027 - 12/10/2015 06:57 PM - naruse (Yui NARUSE)**

- object.c (rb_inspect): dump inspected result with rb_str_escape() instead of raising Encoding::CompatibilityError. [Feature #11801]
- string.c (rb_str_escape): added to dump given string like rb_str_inspect without quotes and always dump in US-ASCII like rb_str_dump.

**Revision 53027 - 12/10/2015 06:57 PM - naruse (Yui NARUSE)**

- object.c (rb_inspect): dump inspected result with rb_str_escape() instead of raising Encoding::CompatibilityError. [Feature #11801]
- string.c (rb_str_escape): added to dump given string like rb_str_inspect without quotes and always dump in US-ASCII like rb_str_dump.

## History

**#1 - 12/10/2015 06:57 PM - naruse (Yui NARUSE)**

*- Status changed from Assigned to Closed*

Applied in changeset r53027.

---

- object.c (rb_inspect): dump inspected result with rb_str_escape() instead of raising Encoding::CompatibilityError. [Feature #11801]
- string.c (rb_str_escape): added to dump given string like rb_str_inspect without quotes and always dump in US-ASCII like rb_str_dump.