Ruby master - Feature #2043
incompatible character encodings
09/04/2009 11:08 PM - vo.x (Vit Ondruch)

<table>
<thead>
<tr>
<th>Status:</th>
<th>Rejected</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority:</td>
<td>Normal</td>
</tr>
<tr>
<td>Assignee:</td>
<td>naruse (Yui NARUSE)</td>
</tr>
<tr>
<td>Target version:</td>
<td>3.0</td>
</tr>
</tbody>
</table>

**Description**

```ruby
begin
Why the following example fails with the "Encoding::CompatibilityError: incompatible character encodings: Windows-1250 and UTF-8" exception?

s = "\u017Elu\u0165ou\u010dk\u00fd"
a = s.encode 'cp1250'
b = s.encode 'utf-8'
c = a + b

I would expect that if the strings are not in the same encoding, that Ruby will do everything they can to satisfy me, but they just tries if there is possible conversion to ASCII otherwise exception is fired. This is really annoying behavior.

Have you considered to allow such string merge?
=end
```

**History**

#1 - 09/05/2009 02:12 AM - naruse (Yui NARUSE)
- Status changed from Open to Assigned
- Assignee set to naruse (Yui NARUSE)

```ruby
begin
Sorry, what is "possible conversion to ASCII"?
=end
```

#2 - 09/07/2009 11:26 AM - yugui (Yuki Sonoda)
- Target version set to 3.0

```ruby
begin
=end
```

#3 - 09/07/2009 03:12 PM - vo.x (Vit Ondruch)

```ruby
begin
In following example, just characters from US-ASCII are used and in this case the addition works well.

s = 'abc'
a = s.encode 'cp1250'
b = s.encode 'utf-8'
c = a + b
=end
```

#4 - 09/07/2009 03:33 PM - naruse (Yui NARUSE)

```ruby
begin
Ruby 1.9 doesn't automatic conversion.
ASCII character set is a special because those characters of ASCII compatible encodings are the same characters.

On Ruby 1.9's view, Unicode is not a superset of Windows-1252.
=end
```

#5 - 09/07/2009 03:39 PM - nobu (Nobuyoshi Nakada)
On Ruby 1.9’s view, Unicode is not a superset of Windows-1252.

Is the “Ruby 1.9’s view” somewhere described in detail? I still have the feeling that it is just half baked :/

People in ISO 8859 may think why Unicode is not a super set of Windows-1252. In Japan, because of lack of standard conversion tables between Japanese legacy encoding (Shift_JIS, EUC-JP, ISO-2022-JP) and Unicode, vendors use different tables. This sad situation made that Unicode is not a simple super set of legacy. Ruby 1.9 inherits this.

If wide consensus for the standard table was made before Ruby 2.0, Ruby 2.0 may have automatic conversion (or Unicode comes to be the internal code).

I'm looking forward Ruby 2.0 and their automatic conversions, since writing \texttt{c = a.encode('utf-8')+ b.encode('utf-8')} to safely concatenate two strings is not sexy at all.