Base Language - Bug #8008

read-json on a 1252 longchar should convert to utf-8 automatically before hydrating temp-tables/datasets, and support other Unicode encodings like UTF-32

11/06/2023 08:32 AM - Tijs Wickardt

Status:	New	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**case_num:**	
vendor_id:	GCD		

Description

History

#1 - 11/06/2023 08:33 AM - Tijs Wickardt

This task should be picked up **after** customer task #7944 is completed.
The customer task #7944 has a special condition: a fix-codepage to "utf-8" with cpinternal=1252.

Task #8008 is specifically about a scenario where OE converts automatically to UTF-8, even without fix-codepage, so no codepage corruption occurs.
FWD should have the same behavior.

1. You need to configure cpinternal=1252 in directory.
2. Then, convert and run the following testcase:

```
def temp-table tt no-undo field c as char.
def var cClob as longchar no-undo.
cClob = substitute('~{"tt":[~{"c":"&1"~}]~}', chr(128,session:charset,"1252")).
temp-table tt:read-json("longchar", cClob, "empty").
find tt.
message "session:charset:" session:charset skip
  "tt.c (euro sign):" tt.c
  view-as alert-box.
```

The desired result is: a valid euro sign.
Currently it shows garbled characters.

#3 - 11/06/2023 08:48 AM - Tijs Wickardt

Additional information:

1. It is expected that cpinternal iso-8859-15 will show the same behavior with the (verbatim) same testcase at #8008-1 , since iso-8859-15 is also a single byte codepage which contains the euro sign. The FWD bug does not manifest itself with cpinternal utf-8 .
2. cpinternal utf-16 and cpinternal utf-32 should be tested as well.
3. In fwd trunk rev 14773 only the fix for #7944 is required. In fwd trunk rev 14790 fixes for both #7944 as well as #8008 are required. See #7944-4 for more details, if needed.

#4 - 12/05/2023 05:51 AM - Tijs Wickardt

- Subject changed from read-json on a 1252 longchar should convert to utf-8 automatically before hydrating temp-tables/datasets to read-json on a 1252 longchar should convert to utf-8 automatically before hydrating temp-tables/datasets, and support other Unicode encodings like UTF-32

In this Note, I summarize the parts from #7944 which were out of scope there.
The convert_json_1252_fcp_memptr.p from #7944 is relevant as well, and after #8008 it should succeed **without** the **fix-codepage** statement. It succeeds in OE 11.7 with cpinternal 1252.

#7944-26

#7944-26

Greg Shah wrote:

> My primary concern is whether OE supports encoding other than UTF-8. The original JSON spec was RFC 4627 and that was replaced by RFC 7159 and then finally by RFC 8259. The older RFCs support other encodings different from UTF-8 even though UTF-8 is the default.
>
> If OE supports other encodings, then we should not hard code it here.

#7944-30

#7944-30

Hi Greg, Guido asked me to take a quick look.
Here is the answer on 'is utf-8 the only codepage needed':
The answer is no.
But for ~~UNDISCLOSED~~, utf-8 is enough.
See code below (header comment: grabbed from OE11.7 help)

```
/* The encoding name must specify a Unicode transformation format.
   Valid values are "UTF-8", "UTF-16", "UTF-16BE", "UTF-16LE", "UTF-32", "UTF-32BE", and "UTF-32LE".
*/
define variable cCp as character no-undo init "UTF-8,UTF-16,UTF-16BE,UTF-16LE,UTF-32,UTF-32BE,UTF-32LE".
define variable cEuro as character no-undo.
define variable iCp as integer no-undo.
define temp-table tt no-undo field c as char.
cEuro = chr(128,session:charset,"1252").
create tt.
tt.c = cEuro.
do iCp = 1 to num-entries(cCp):
  temp-table tt:write-json("file", "tijs_json_encode.json", true, entry(iCp,cCp)).
  temp-table tt:read-json("file", "tijs_json_encode.json", "empty").
  find tt.
  if tt.c <> cEuro then undo, throw new Progress.Lang.AppError(substitute("Error: tt.c &1 <> cEuro &2"
, tt.c, cEuro)).
end.
message tt.c view-as alert-box.
```

This testcase can be run in any single or multi-byte cpinternal.
However, on single byte codepages, the euro sign is only shown at (for example) 1252 and iso-8859-15. In other single byte codepages a literal question mark is substituted, this is by design.

#7944-36

#7944-36

Ovidiu Maxiniuc wrote:

> Thanks to Greg's investigations, we can see that although the latest standard encoding is UTF-8, in previous RFCs it was only the default and recommended CP.
>
> The way I see the problem is that OE is adhering to old standards, or just being lenient in this case (or back compatible).
>
> So, when a JSON is **saved** in 4GL, the CP used for encoding are constrained to the list from #7944-30. This is the parameter of write-json. However, when we **read** the file, the read-json method does not offer the option to specify the CP. This is normal. The encoding 'is' already in the source byte stream, not an option for the caller. The stream reader should be smart enough to detect the encoding based on the analysis of the first few bytes of the stream, and in no case should we enforce UTF-8 encoding. For an example of how to do this, please see rfc4627, paragraph §3. Encoding. I am not sure whether the InputStreamReader is able to properly detect the CP, but if it does not (since the BOM is not present), we need detects the encoding using other means.