Base Language - Feature #12193

ROMAN-8 codepage support

11/16/2026 03:49 PM - Greg Shah

Status:	New	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Ovidiu Maxiniuc	**% Done:**	90%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**reviewer:**	Constantin Asofiei
vendor_id:	GCD	**production:**	No
case_num:		**env_name:**	
version_reported:		**topics:**	
version_resolved:			

Description

History

#1 - 11/16/2026 03:50 PM - Greg Shah

Problem

OpenEdge supports the HP Roman-8 codepage, named ROMAN-8 in convmap.cp. A project uses it to exchange XML and text with an external system:

```
DEFINE VARIABLE lcXml AS LONGCHAR NO-UNDO.

FIX-CODEPAGE(lcXml) = "ROMAN-8".
hDoc:SAVE("LONGCHAR", lcXml).

COPY-LOB FROM FILE cInFile TO lcXml CONVERT SOURCE CODEPAGE "ROMAN-8".
```

FWD lists ROMAN-8 in I18nOps, but maps it to a Java charset name that does not exist ("support is undefined"). The JDK has no HP Roman-8 charset. Any conversion to or from ROMAN-8 therefore fails or falls back, depending on the operation.

Work

1. Add a Java Charset for HP Roman-8, built from the OpenEdge ROMAN-8 table in convmap.cp (both directions, including how OpenEdge handles code points that have no mapping), and register it so it can be found by name.
2. Map the OpenEdge name ROMAN-8 to it in I18nOps, and accept the usual aliases (hp-roman8, roman8) where OpenEdge does.
3. Check every place a codepage name is accepted: FIX-CODEPAGE, CODEPAGE-CONVERT, COPY-LOB ... CONVERT, INPUT FROM / OUTPUT TO ... CONVERT, X-DOCUMENT:SAVE and LOAD with an explicit encoding, READ-XML / WRITE-XML encodings, and GET-CODEPAGE / IS-CODEPAGE-FIXED results.
4. Unmappable characters: reproduce the OpenEdge result (substitution character or error, as OpenEdge does for this table).
5. Update the gap marking for the codepage.

Testcases

- Round trip of every code point 0-255 between ROMAN-8 and UTF-8 and ISO8859-1, compared with Windows or Linux OpenEdge.
- A character with no mapping in each direction, compared with OpenEdge.
- X-DOCUMENT:SAVE to a LONGCHAR fixed to ROMAN-8, and LOAD back, with non-ASCII content.
- COPY-LOB and stream CONVERT with ROMAN-8 as source and target.

#2 - 11/19/2026 08:55 PM - Ovidiu Maxiniuc

- % Done changed from 0 to 90

- Assignee set to Ovidiu Maxiniuc

- reviewer Constantin Asofiei added

I have created branch 12193a and committed the implementation I did a couple of weeks ago. Committed revision 16865 (rebased from 16862).

The basic support is composed of the Roman8Charset which implements the actual mapping (and the encoder and decoders) and the FwdCharsetProvider. The latter may accommodate other CPs, in the future. Note that the build.gradle takes care that the new files are packaged into fwdspi.jar instead of standard p2j.jar.

In I18nOps.java we already have:

```
convmap2JavaDefault.put("ROMAN-8", "ROMAN-8"); // support is undefined
```

Based on original lines from 1st note, I created the following testcase:

```
output to \tmp\12193a.txt.

DEFINE VARIABLE lcXml   AS LONGCHAR  NO-UNDO.
DEFINE VARIABLE hDoc    AS HANDLE    NO-UNDO.
DEFINE VARIABLE hRoot   AS HANDLE    NO-UNDO.
DEFINE VARIABLE hText   AS HANDLE    NO-UNDO.
DEFINE VARIABLE cInFile AS CHARACTER NO-UNDO.
DEFINE VARIABLE cText   AS CHARACTER NO-UNDO.
DEFINE VARIABLE cBytes  AS CHARACTER NO-UNDO.
DEFINE VARIABLE mBytes  AS MEMPTR    NO-UNDO.
DEFINE VARIABLE i       AS INTEGER   NO-UNDO.

/* Non-ASCII text, built from the ROMAN-8 byte values so that the source stays
   ASCII and the result does not depend on -cpinternal:
   "caf<e-acute> <u-umlaut> <pound> <sharp-s> <c-cedilla>" */
cText = "caf" + CHR(197, SESSION:CPINTERNAL, "ROMAN-8")
      + " "   + CHR(207, SESSION:CPINTERNAL, "ROMAN-8")
      + " "   + CHR(187, SESSION:CPINTERNAL, "ROMAN-8")
      + " "   + CHR(222, SESSION:CPINTERNAL, "ROMAN-8")
      + " "   + CHR(181, SESSION:CPINTERNAL, "ROMAN-8").

/* 1. X-DOCUMENT saved into a LONGCHAR fixed to ROMAN-8. */
CREATE X-DOCUMENT hDoc.
CREATE X-NODEREF  hRoot.
CREATE X-NODEREF  hText.

hDoc:CREATE-NODE(hRoot, "root", "ELEMENT").
hDoc:APPEND-CHILD(hRoot).
hDoc:CREATE-NODE(hText, ?, "TEXT").
hText:NODE-VALUE = cText.
hRoot:APPEND-CHILD(hText).

FIX-CODEPAGE(lcXml) = "ROMAN-8".
hDoc:SAVE("LONGCHAR", lcXml).

MESSAGE "SAVE codepage:" GET-CODEPAGE(lcXml) SKIP
        "text found:"    (INDEX(lcXml, cText) > 0) SKIP
        "chars/bytes:"   LENGTH(lcXml, "CHARACTER") "/" LENGTH(lcXml, "RAW") SKIP
        STRING(lcXml).

DELETE OBJECT hText.
DELETE OBJECT hRoot.
DELETE OBJECT hDoc.

/* 2. File holding the same text as raw ROMAN-8 bytes, read back with conversion. */
cInFile = SESSION:TEMP-DIRECTORY + "p_12193-a.txt".
cBytes  = "99,97,102,197,32,207,32,187,32,222,32,181".

SET-SIZE(mBytes) = NUM-ENTRIES(cBytes).
DO i = 1 TO NUM-ENTRIES(cBytes):
   PUT-BYTE(mBytes, i) = INTEGER(ENTRY(i, cBytes)).
END.
COPY-LOB FROM mBytes TO FILE cInFile.
```

```
SET-SIZE(mBytes) = 0.

COPY-LOB FROM FILE cInFile TO lcXml CONVERT SOURCE CODEPAGE "ROMAN-8".

MESSAGE "COPY-LOB codepage:" GET-CODEPAGE(lcXml) SKIP
        "equal:"             (lcXml = cText) SKIP
        "chars/bytes:"       LENGTH(lcXml, "CHARACTER") "/" LENGTH(lcXml, "RAW") SKIP
        STRING(lcXml).

OS-DELETE VALUE(cInFile).
```

It will print

```
SAVE codepage: ROMAN-8
 text found: yes
 chars/bytes: 63 / 63
 <?xml version="1.0" encoding="UTF-8"?><root>café ü £ ß ç</root>
COPY-LOB codepage: ROMAN-8
 equal: yes
 chars/bytes: 12 / 12
 café ü £ ß ç
```

in Java and

```
SAVE codepage: ROMAN-8
text found: yes
chars/bytes: 48 / 48
<?xml version="1.0" ?>
<root>café ü ° ß ç</root>
COPY-LOB codepage: ROMAN-8
equal: yes
chars/bytes: 12 / 12
café ü ° ß ç
```

in 4GL. There is something wrong here, which is work in progress. The 187 (0xBB) is £ according to https://en.wikipedia.org/wiki/HP_Roman map. The output of 4GL (°) can be seen as 0xB3.