CSYSTEMS™

PROGRAMSKI PAKET ZA KNJIGOVODSTVO

COBA Systems

UNICODE
for Alaska Xbase++ Applications

15.10.2020

Xbase++ and Unicode character set

1. Xbase ++ can work with **ANSI code page 1250 Serbian Latin** (charset value 238, hex xEE) in which case the Xbase ++ application will correctly display the Serbian Latin ANSI character set and the letters **ŠĐČĆŽ** in Latin (charset name EE_CHARSET). Serbian Latin letters must be entered in the source code from the ANSI code page 1250 of the editor from a keyboard operating in Serbian Latin mode. Windows should be localized to the Serbian Latin locale set. When entering in the editor, the letters will be displayed correctly.

2. Xbase ++ can work with **ANSI code page 1251 Serbian Cyrilic** (charset value 204, hex xCC) in which case the Xbase ++ application will correctly display the Serbian Cyrilic ANSI character set and the letters **ШЂЧЋЖ** in Cyrillic (charset name RUSSIAN_CHARSET). Serbian Cyrilic letters must be entered into the source code from the ANSI code page 1250 editor from a keyboard operating in Serbian Cyrilic mode. Windows should be localized to the Serbian Latin locale set. When entering the editor, the letters will not be displayed correctly.

3. The Xbase ++ application can display and print both Serbian Latin and Serbian Cyrilic letters, obtained in the manner described above, if the font set to oFont is used for Serbian Latin **oFont:CodePage := 238** and if for Serbian Cyrilic letters use the font set to **oFont:CodePage := 204**.

4. Everything said for code pages 1250 and 1251 also applies to other code pages, ie to other local languages in which Xbase ++ applications are written and executed.

5. If Xbase ++ would support working with a **unicode character set**, then both Latin and Cyrillic would have to be written to the source code via one of the unicode editors, without taking into account the code pages, and the editor would convert everything written to unicode letters. The Xbase ++ compiler would compile everything written as unocod letters and the Xbase ++ application would have unicod letters that would display and print both Latin and Cyrillic or anything else at the same time during execution. But Xbase ++ cannot use unicod letters and for now different character sets can be used within one application only through the technique described in points 1, 2, 3 and 4.

Xbase ++ cannot work with a UTF-8 or UTF-16 character set because then the Xbbase ++ application cannot decode and display any UTF-8 and UTF-16 letters, but displays illegible characters (hooks and chains)

Do Xbase++ applications need UTF-8 or UTF-16 letters at all? The answer is: She still doesn't need them to the extent that you can't do without them. For now, what the Xbase++ application has is enough.

The question is: When and for what are UNICODE characters used in the Xbase++ application?

The answer is:

Only in two cases:

- when the Xbase++ Application from other applications downloads strings and texts made there with UNICODE characters, and should be used in the Xbase++ application - of course after translating unicode to ANSI 1250 or 1251

Unicode2Str(cUtf16) -> cAnsi for UTF-16 LE

cUtf8ToAnsi (cUtf8) -> cAnsi for UTF-8

- when the Xbase++ Application translates its strings and texts from ANSI 1250 and 1251 into UNICODE and sends them to other applications that can use these strings and texts only as UNICODE. Translation from ANSI 1250 and 1251 to unicode goes with:

Str2Unicode(cAnsi) -> cUtf16 for UTF-16 LE

cAnsiToUtf8(cAnsi) -> cUtf8 for UTF-8

What to use in creating Xbase++ Application for working with unicode character set

Alaska Xbase ++ supports UTF-16 LE and does not support UTF-8, so the following Xbase ++ functions do not work with UTF-8:

```
IsUnicode(string)   -> true/false
Str2Unicode(cAnsi)  -> cUtf16
Unicode2Str(cUtf16) -> cAnsi
```

For UTF-8, the ot4Xb.dll / lib library (https://www.xbwin.com) must be used, which is an integral part of the Xbase++ runtime package since version 2.0, with the following functions:

```
cAnsiToUtf8( cAnsi ) -> cUtf8
cUtf8ToAnsi(cUtf8)   -> cAnsi
cOemToUtf8(cOem)     -> cUtf8
cUtf8ToOem(cUtf8)    -> cOem
cSzAnsi2Wide( <cAnsi> ) -> cUtf8
cSzWide2Ansi( <cUtf8> ) -> cAnsi
```

COBA Systems Library CSYSTEMSU.DLL / LIB

used for UTF-8 for Serbian Latin and Serbian Cyrilic

ANSI_lat_cyr: = UTF8_TO_Cyrilic1251_Latin1250 (UTF8string)

From unicode text translates Latin into Latin ANSI 1250 and Cyrillic into Cyrillic ANSI 1251

ANSI_lat: = UTF8_TO__Latin1250 (UTF8string)

From unicode text translates Latin into Latin ANSI 1250 and Cyrillic into Latin ANSI 1250

ANSI_cyr: = UTF8_TO__Cyrilic1251 (UTF8string)

From unicode text translates Latin into Cyrillic ANSI 1251 and Cyrillic into Cyrillic ANSI 1251

Text editor - source code

If it is necessary for the source code for Xbase ++ applications to be written in a unicode character set (which is not necessary for now), you should know that text editors recognize the unicode character set by the BOM tag and write this tag at the beginning of each unicode text.

BYTE ORDER MARK A byte order mark (BOM) is a special Unicode character (U + FEFF) at the beginning of each text or file that contains unicode text that signals several things to a program that reads text

BOM use is optional. Its presence interferes with the use of UTF-8 by software that does not expect non-ASCII bytes at the beginning of the file, but which could otherwise process the text stream.

UTF-8 BOM signature at the begining of the file:

```
? Chr(0xEF) + Chr(0xBB) + Chr( 0xBF )
```

The **byte order mark** (**BOM**) is a particular usage of the special Unicode character, U+FEFF BYTE ORDER MARK, whose appearance as a magic number at the start of a text stream can signal several things to a program reading the text:[1]

- The byte order, or endianness, of the text stream in the cases of 16-bit and 32-bit encodings;
- The fact that the text stream's encoding is Unicode, to a high level of confidence;
- Which Unicode character encoding is used.

BOM use is optional. Its presence interferes with the use of UTF-8 by software that does not expect non-ASCII bytes at the start of a file but that could otherwise handle the text stream.

The Xbase ++ compiler cannot read and decode the BOM tag, so the BOM must be removed from the unicode text file (source code) to be sent to the compiler. Or a unicode text editor must be used for unicode UTF-8 without a BOM tag. Unicode UTF-16 LE and UTF-16 BE always have a BOM so that editors can distinguish them from each other.

So, if you need to write Xbase ++ source code that generates unicode letters, then it is best to use a text editor that works in UTF-8 mode without BOM.

Write unicode source code

Dok je editor teksta u UTF-8 modu rada svako slovo serbian latin ŠĐČĆŽ biće kodirano kao UTF-8 a ne kao ANSI 1250 pa će se u Xbase++ aplikaciji prikazivati kao UTF-8 nečitljiv znak, osim ako se ne prevede komandom: **cUtf8ToAnsi(UTF8string)**

While the text editor is in UTF-8 mode, each letter of the Serbian Latin ŠĐČĆŽ will be encoded as UTF-8 and not as ANSI 1250, so in the Xbase++ application it will be displayed as a UTF-8 illegible character, unless translated with the command: **cUtf8ToAnsi (UTF8string)**

So, it is unnecessary to encode for Xbase++ applications in the editor as UTF-8 without BOM. Nothing was gained by that, only problems. If the Xbase++ application should have both text with Latin letters and text with Cyrillic letters, it should be written from the ANSI editor that works on code page 1250 (or windows are set to set locale code page 1250) which will make all Latin, including **ŠĐČĆŽ**, correct by default, and Cyrillic texts should be coded with the Cyrillic keyboard included, whereby hooks and chains will be obtained (the user will not see the Cyrillic alphabet) which will later be displayed in the application as Cyrillic only with

```
oFont:CodePage := 204
```

The reverse is true for setting windows to the ANSI 1251 code page, which correctly shows Cyrillic in the editor, but not Latin, and from the application, Latin can only be displayed with:

oFont:CodePage := 238

Example:

Write this string in the UTF-8 text editor:

1. EDITOR_UTF8_string := 'Lat = ŠĐČĆŽ Cyr = ШЂЧЋЖ'

Here the editor needs to be in UTF-8 mode only to be able to display each written character in Latin and Cyrillic or any other alphabet so that the editor user sees that character correctly. The editor saves the source code and UTF-8 characters in a text file.

The Xbase compiler translates this as UTF 8 and the application receives the EDITOR_UTF8_string variable as a UTF-8 string and displays it illegibly (displays double-byte UTF-8 characters).

2. UTF8string := EDITOR_UTF8_string

This UTF8string must now be commanded from Xbase++

cUtf8ToAnsi(UTF8string)

translate into what Xbase ++ can display legibly and that is:

ANSI code page 1250 (238) Serbian Latin Character set,

or

ANSI code page 1251 (204) Serbian Cyrilic Character set.

But Xbase can only show this if:

If **SET CHARSET TO ANSI** is set in the application

- if the font is set to

oFont:CodePage := 238 will display Serbian Latin

- if the font is set to

oFont:CodePage := 204 will display Serbian Cyrilic

- If windows is set to Set Locale Serbian Latin

then no need to ask

oFont:CodePage := 238

because all Serbian Latin **ŠĐČĆŽ** will be displayed correctly

but must be specified to display Cyrillic

oFont:CodePage := 204

- If windows is set to Set Locale Serbian Cyrilic

then no need to ask

oFont:CodePage := 204

because all Serbian Cyrilic **ШЂЧЋЖ** will be displayed correctly

but it must be specified to display the Latin alphabet

oFont:CodePage := 238

```
// ot4Xb.dll

  3. ANSIstring1a := cSzWide2Ansi(UTF8string)
  4. ANSIstring1b := cUtf8ToAnsi(UTF8string)

// CSYSTEMSU.DLL

  5. ANSIstring2  := UTF8_TO_Cyrilic1251_Latin1250(UTF8string)
  6. ANSIlat      := UTF8_TO__Latin1250(UTF8string)
  7. ANSIcyr      := UTF8_TO__Cyrilic1251(UTF8string)

  8. MsgBox(;
     "EDITOR string UTF8 = " + EDITOR_utf8_string +cr+;
     "APPLICATION string UTF8  = " + UTF8string +cr+;
     "TRANSLATE--------------" +cr+;
     "ANSI string ot4Xb.dll = " + ANSIstring1a +cr+;
     "ANSI string ot4Xb.dll = " + ANSIstring1b +cr+;
     "TRANSLATE--------------" +cr+;
     "ANSI CSYSTEMS lat-cyr = " + ANSIstring2 +cr+;
     "ANSI CSYSTEMS lat     = " + ANSIlat +cr+;
     "ANSI CSYSTEMS cyr     = " + ANSIcyr +cr+;
     "",”TEST”)
```

Code:

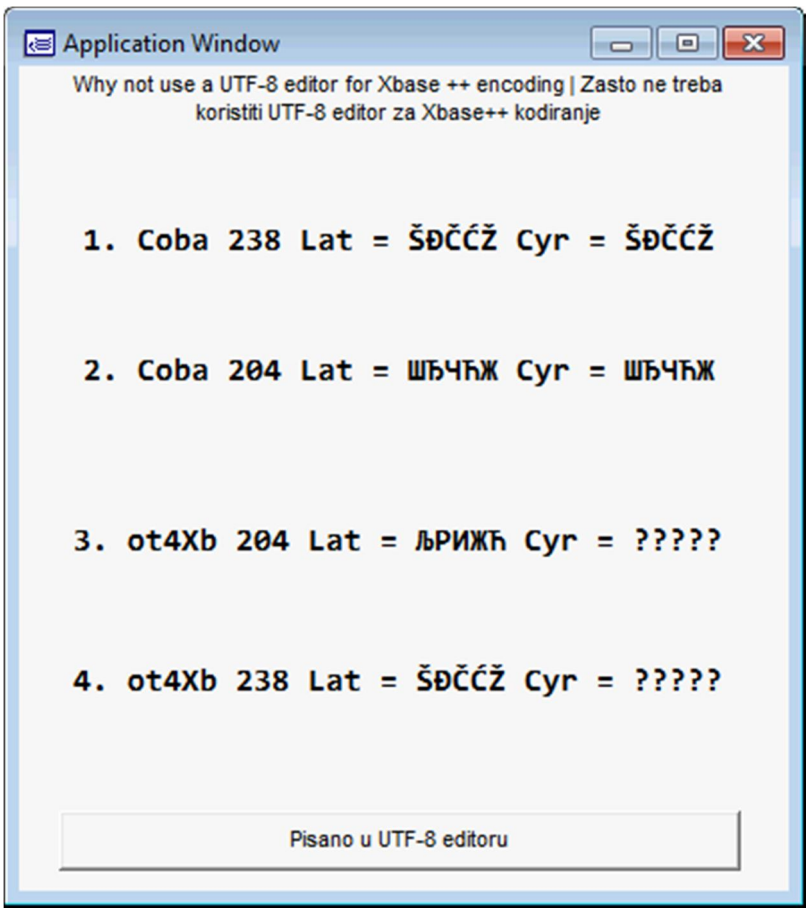
```
#include "appevent.ch"
#include "xbp.ch"
#include "common.ch"
#include "Font.ch"
#include "Gra.ch"

#pragma library("ot4Xb.lib")
#pragma library("CSYSTEMSU.LIB") // options
*************************************************************************
PROCEDURE TEST() // Program: COBA_UTF8_EDITOR.EXE
*************************************************************************

   LOCAL oDlg, aPos[2], aSize[2], oXbp // ...
   ********************
   SET CHARSET TO ANSI  // THIS IS MANDATORY // OVO JE OBAVEZNO !
   ********************

   aSize[1]      := 400
   aSize[2]      := 400
   aPos[1]       := 0.5 * aSize[1]
   aPos[2]       := 0.5 * aSize[2]

   oDlg          := XbpDialog():new( ,, aPos, aSize )
   oDlg:title    := "Application Window"
   oDlg:taskList := .T.
   oDlg:close    := {|| PostAppEvent( xbeP_Quit ) }
   oDlg:create()
   oDlg:drawingArea:setFontCompoundName( FONT_HELV_SMALL )
   SetAppWindow( oDlg )
   SetAppFocus( oDlg )

EDITOR_UTF8_string := 'Lat = ŠĐČĆŽ Cyr = ШЂЧЋЖ'
// uključiti Serbian Latin tastaturu i sa nje ispisati latinična slova
// uključiti Serbian Cyrilic tastaturu i sa nje ispisati ćirilična slova
UTF8string := EDITOR_UTF8_string  // dobije se UNICODE string

// ot4Xb.dll
  ANSIstring1a := cSzWide2Ansi(UTF8string) //
  ANSIstring1b := cUtf8ToAnsi(UTF8string)

// CSYSTEMSU.DLL - COM1251.PRG
  ANSIstring2  := UTF8_TO_Cyrilic1251_Latin1250(UTF8string)
  ANSIlat      := UTF8_TO__Latin1250(UTF8string)
  ANSIcyr      := UTF8_TO__Cyrilic1251(UTF8string)

// da bi se videla latinica mora se setovati Latin font objekt
// na codepage 238
// "LATIN SERBIAN (font object)
oFont1 := Fo_lat(12,"Consolas")
// da bi se videla ćirilica mora se setovati Cyrilic font objekt
```

```
// na codepage 204
// "CYRILIC SERBIAN (font object)
oFontc := Fo_cyr(12,"Consolas")
//----------

   oXbp         := XbpStatic():new( oParent,, {20,350}, {340,60} )
   oXbp:caption := "Why not use a UTF-8 editor for Xbase ++ encoding |"+;
                   "Zasto ne treba koristiti UTF-8 editor za Xbase++ kodiranje"
   oXbp:options := XBPSTATIC_TEXT_CENTER + XBPSTATIC_TEXT_WORDBREAK
   oXbp:create()

// Napomena:
// kod XbpPushbutton() ova tehnika ne funkcioniše isprtavno
// Note:
// with XbpPushbutton () this technique does not work properly

************************************************* 1
* UTF-8 translate in ANSI 1025 (238)

   oXbp         := XbpStatic():new( oParent,, {20,305}, {340,30} )
   oXbp:caption := "1. Coba 238 "+ANSIlat
   oXbp:options := XBPSTATIC_TEXT_CENTER
   oXbp:create()
   oXbp:setFont(oFontL) // Serbian Latin

/*
   oXbp          := XbpPushbutton():new( oParent, , {20,280}, {340,30} )
   oXbp:caption  := "1.a Coba 238 "+ANSIlat // UTF8string
   oXbp:activate := {|| NIL }
   oXbp:create()
   oXbp:setFont(oFontL)
*/
************************************************

************************************************* 2
* UTF-8 translate in ANSI 1251 (204)

   oXbp         := XbpStatic():new( oParent,, {20,240}, {340,30} )
   oXbp:caption := "2. Coba 204 "+ANSIcyr
   oXbp:options := XBPSTATIC_TEXT_CENTER
   oXbp:create()
   oXbp:setFont(oFontC) // Serbian Cyrilic

 /*
   oXbp          := XbpPushbutton():new( oParent, , {20,215}, {340,30} )
   oXbp:caption  := "2.a Coba 204 "+ANSIcyr
   oXbp:activate := {|| NIL }
   oXbp:create()
   oXbp:setFont(oFontC)
*/
************************************************

************************************************* 3
* UTF-8 translate in ANSI 1251 (204)
```

```
   oXbp         := XbpStatic():new( oParent,, {20,155}, {340,30} )
   oXbp:caption := "3. ot4Xb 204 "+ANSIstring1b
   oXbp:options := XBPSTATIC_TEXT_CENTER
   oXbp:create()
   oXbp:setFont(oFontC) // Serbian Cyrilic

/*
   oXbp          := XbpPushbutton():new( oParent, , {20,130}, {340,30} )
   oXbp:caption  := "3.a ot4Xb 204 "+ANSIstring1b
   oXbp:activate := {|| NIL }
   oXbp:create()
   oXbp:setFont(oFontC)
*/
************************************************

************************************************* 4
* UTF-8 translate in ANSI 1250 (238)

   oXbp         := XbpStatic():new( oParent,, {20,85}, {340,30} )
   oXbp:caption := "4. ot4Xb 238 "+ANSIstring1b // ANSIcyr
   oXbp:options := XBPSTATIC_TEXT_CENTER
   oXbp:create()
   oXbp:setFont(oFontL) // Serbian Latin

/*
   oXbp          := XbpPushbutton():new( oParent, , {20,60}, {340,30} )
   oXbp:caption  := "4.a ot4Xb 238 "+ANSIstring1b
   oXbp:activate := {|| NIL }
   oXbp:create()
   oXbp:setFont(oFontL)
*/
************************************************

   oXbp          := XbpPushbutton():new( oParent, , {20,10}, {340,30} )
   oXbp:caption  := "UTF-8 EDITOR"
   oXbp:activate := {|| PostAppEvent( xbeP_Quit ) }
   oXbp:create()

   DO WHILE nEvent <> xbeP_Quit
      nEvent := AppEvent( @mp1, @mp2, @oXbp )
      oXbp:handleEvent( nEvent, mp1, mp2 )
   ENDDO

RETURN // PROCEDURE TEST


************************************************************
FUNCTION Fo_lat(nFsize,cfname)
************************************************************
* ovo je po defaultu za setup windowsa na Serbian Latin pa se
* tada ne mora (kao ovde) setovati font na codepage 238
* "LATIN SERBIAN (font object)
*-------------------------------
oFontl := XbpFont():new()
oFontl:familyName := cFname // 'Consolas' //'Arial'
```

```
oFontl:nominalPointSize := nFsize // 14
oFontl:bold := .t.
//oFontl:italic := .t.
//oFontl:underScore := .t.
oFontl:codePage := 238 // CodePage Serbian latin=238/1250
oFontl:create() // L A T I N I C A
RETURN oFontL

*****************************************************************
FUNCTION Fo_cyr(nFsize,cfname)
*****************************************************************
* da bi se videla ćirilica mora se setovati Cyrilic font objekta
* na codepage 204
* "CYRILIC SERBIAN (font object)
*-------------------------------
oFontc := XbpFont():new()
oFontc:familyName := cFname //'Consolas' // 'Arial'
oFontc:nominalPointSize := nFsize //14
oFontc:bold := .t.
//oFontc:italic := .t.
//oFontc:underScore := .t.
oFontc:codePage := 204 // CodePage Serbian cyrilic=204/1251
oFontc:create() // Ć I R I L I C A
RETURN oFontC
```

Program: COBA_UTF8_EDITOR.EXE



Help:

Since it is a unicode string THAT HAS LETTERS IN LATIN AND CYRILLIC

"Lat = ŠĐČĆŽ Cyr = ЩЂЧЋЖ"

written from UTF-8 editor, then the translation of that string via the function cUtf8ToAnsi (UTF8string) gave correct Latin (Figure 4) and incorrect-canceled Cyrillic (Figure 3).

The translation of the string via the function UTF8_TO__Latin1250 (UTF8string) gave he converted the correct Latin and Cyrillic into the correct Latin (Figure 1)

The translation of the string via the function UTF8_TO__Cyrilic1251 (UTF8string) gave Latin translated into Cyrillic and correct Cyrillic (Figure 2)

That Latin and Cyrillic are written as ANSI 1250 string from ANSI editor
then that ANSI 1250 string would be via the functions Fo_lat (nFsize, cfname) and Fo_cyr (nFsize, cfname) was translated into the correct letters BOTH LATIN AND CYRILLIC