CET BASIC Printer Extensions
W32 App Builder Version

Class file CBEPrint.dll

Classes

- Printer
  Class printer provides simplified access to Windows print spooler

Class Printer

Line buffer and word wrapping

Printer object keeps all text posted to object through **Text** or **TextAt** calls in line buffer. Line buffer is printed when **NewLine** call is performed, or \n is encountered in input text.
If length of text to be printed is longer than page width, printer object destroys the end of the line or wraps line to beginning of the next line. Wrapping is performed on word boundary. Word wrapping mode can be set by **WordWrap** call. Default value for **WordWrap** property is 0 for absolute mode and 1 for virtual mode.

Modes

There are two printing modes – absolute mode and virtual mode. Mode is set by **Create** call and cannot be changed for created printer object.
In virtual mode some printer calls works in coordinates related to current **virtual page.** Virtual page coordinates can be set by **SetPage** call.
The following calls have special behaviour in virtual mode:

1. **Text, TextAt, NewPosition, Position** calls works with coordinates related to current virtual page
2. Default value for **WordWrap** property is 1
3. Printer object does not go to new physical page authomatically. Instead, when print line exeeds current virtual page, object keeps rest of line in line buffer (up to first “\n”). Application program must check line buffer after every **NewLine** call or after **Text** call if text contain “\n”. This check can be done with **IsEmpty** call. If line buffer is not empty, it means that virtual page is full. Application program must set new virtual page, on the same physical page, or call **NewPage,** then set virtual page on next physical page and call **NewLine** again to print rest of the line kept in the line buffer.
4. Printer objectd discards all input line content after first “\n” sequence. This restriction will be eliminated in next releases.

Methods

Create (Name, Mode) create object, do not start print job

Name (String) - printer name to be created, open dialog box if name is empty ("").
Mode (String) - printer open modes:
“” – normal mode
“Virtual” – virtual mode

CALL call:
CALL cetObjectCreate(ADDROF(obj%), “Printer”, name$, mode$, 2)
MACRO call:
PrinterCreate(obj%, name$, mode$)

CreateCommon (Name, Mode) create common object, do not start print job

Name (String) - printer name to be created, open dialog box if name is empty ("").
Mode (String) - printer open modes:

CALL call:
CALL cetObjectCreateCommon(ADDROF(obj%), “Printer”, name$, mode$, 2)
MACRO call:
PrinterCreateCommon(obj%, name$, mode$)

Destroy() destroy object, cancel unfinished print job

CALL call:
CALL catObjectDestroy(obj%)
MACRO call:
PrinterDestroy(obj%)

Begin (Name) start document. Every document is separated print job for print spooler. Print spooler guaranties that document will not be mixed with other documents

Name (String) - document name. Document name appears in Printer Status window.

CALL call:
CALL cetObjectExecute(obj%, “Begin”, name$, 1)
MACRO call:
PrinterBegin(obj%, name$)

End () end document

CALL call:
CALL cetObjectExecute(obj%, “End”, 0)
MACRO call:
PrinterEnd(obj%)

Cancel () cancel document

CALL call:
CALL cetObjectExecute(obj%, “Cancel”, 0)
MACRO call:
PrinterCancel(obj%)

Font (Typeface, Size, Style) set font for text output

Typeface (String) - font name
Size (Integer) - font size
Style (String) - one or more font styles - B[old], I[talic], [U]nderline

CALL call:
CALL cetObjectExecute(obj%, “Font”, typeface$, size%, style$, 3)

MACRO call:
PrinterFont(obj%, typeface$, size%, style$)

Text (String) output text string starting from current cursor position

String - text string can contain following escape (not control!) characters:
\\ - backslash
\n - end of line
\f - end of page
\r - right alignment
\l - left alignment
\c - center alignment
\j - justification
\0..9 - select font

CALL call:
CALL cetObjectExecute(obj%, “Text”, string$, 1)
MACRO call:
PrinterText(obj%, string$)

TextAt (Position,String) output text string starting from current cursor position

Position – pixel possition
String - text string can contain following escape (not control!) characters:
\\ - backslash
\n - end of line
\f - end of page
\0..9 - select font

CALL call:
CALL cetObjectExecute(obj%, “Text”, string$, 1)
MACRO call:
PrinterText(obj%, string$)

NewLine() go to beginning of the next line

CALL call:
CALL cetObjectExecute(obj%, “NewLine”, 0)
MACRO call:
PrinterNewLine(obj%)

NewLine(Position) go to beginning of line at given vertical offset

Position (Integer) - new vertical position

CALL call:
CALL cetObjectExecute(obj%, “NewLine”, position%, 1)
MACRO call:
PrinterNewPosition(obj%, position%)

NewPage() new page

CALL call:
CALL cetObjectExecute(obj%, “NewPage”, 0)

MACRO call:
PrinterNewPage(obj%)

Left () left text align

CALL call:
CALL cetObjectExecute(obj%, “Left”, 0)
MACRO call:
PrinterLeft(obj%)

Right () right text align

CALL call:
CALL cetObjectExecute(obj%, “Right”, 0)
MACRO call:
PrinterRight(obj%)

Center () center text

CALL call:
CALL cetObjectExecute(obj%, “Center”, 0)
MACRO call:
PrinterCenter(obj%)

Justify () justify text

CALL call:
CALL cetObjectExecute(obj%, “Justify”, 0)
MACRO call:
PrinterJustify(obj%)

SetPage (Left, Top, Width, Height) set virtual page position and size
If requested virtual page doesn’t fit into physical page, or some of coordinates <0, virtual page is set to physical page.

Left (Integer) – absolute position of left top corner in pixels
Top (Integer) – absolute position of left top corner in pixels
Width (Integer) – virtual page width in pixels
Height (Integer) – virtual page height in pixels

CALL call:
CALL cetObjectExecute(obj%, “VirtPage”, left%, top%, width%, height%, 4)
MACRO call:
PrinterSetPage(obj%, left%, top%, width%, height%)
PrinterResetPage(obj%)

DrawLine (FromX, FromY, ToX, ToY, Width, Style) Draw line between two points

FromX (Integer) – absolute horizontal position of start point
FromY(Integer) – absolute vertical position of start point
ToX (Integer) – absolute horizontal position of end point
ToY (Integer) – absolute vertical position of end point
Width(Integer) – line width in pixels
Style(Integer) – line style:
PrinterLineSolid

PrinterLineDash
PrinterLineDot
PrinterLineDashDot
PrinterLineDashDotDot

CALL call:
CALL cetObjectExecute(obj%, “Line”, fromX%, fromY%, toX%, toY%, width%, style%, 6)
MACRO call:
PrinterDrawLine(obj%, fromX%, fromY%, toX%, toY%, width%, style%)

DrawBox (Left, Top, Width, Height, LineWidth, Style) Draw rectangular box

Left (Integer) – absolute horizontal coordinate of top left corner
Top(Integer) – absolute vertical coordinate of top left corner
Width (Integer) – box width
Heigth (Integer) – box height
LineWidth(Integer) – line width in pixels
Style(Integer) – line style:
PrinterLineSolid
PrinterLineDash
PrinterLineDot
PrinterLineDashDot
PrinterLineDashDotDot

CALL call:
CALL cetObjectExecute(obj%, “Box”, left%, top%, width%, height, lwidth%, style%, 6)
MACRO call:
PrinterDrawBox(obj%, left%, top%, width%, height, lwidth%, style%)

DrawBitmap (Name,Left, Top, Width, Height) Draw bitmap from .bmp file
Bitmap size is adjusted to given **Width** and **Height**

Name(String) – bitmap file name and path (Windows, not THEOS name)
Left (Integer) – absolute horizontal coordinate of top left corner
Top(Integer) – absolute vertical coordinate of top left corner
Width (Integer) – bitmap width
Heigth (Integer) – bitmap height

CALL call:
CALL cetObjectExecute(obj%, “Bitmap”, name$, left%, top%, width%, height, 5)
MACRO call:
PrinterDrawBitmap(obj%, name$, left%, top%, width%, height)

Properties

Status (Integer, RO) current status of printer object
0 - Ok,
>0 - Error

CALL call:
CALL cetObjectGetProperty(obj%, “Status”, ADDROF(status%))

MACRO call:
PrinterStatus(obj%, status%)

PageWidth(Integer, RO) page width in device units (pixels)

CALL call:
CALL cetObjectGetProperty(obj%, “PageWidth”, ADDROF(width%))
MACRO call:
PrinterPageWidth(obj%, width%)

PageLength(Integer, RO) page length in device units (pixels)

CALL call:
CALL cetObjectGetProperty(obj%, “PageLength”, ADDROF(length%))
MACRO call:
PrinterPageLength(obj%, length%)

Position(Integer, RO) current relative (virtual page) vertical position in device units (pixels)

CALL call:
CALL cetObjectGetProperty(obj%, “Position”, ADDROF(pos%))
MACRO call:
PrinterPosition(obj%, pos%)

AbsPosition(Integer, RO) current absolute vertical position in device units (pixels)

CALL call:
CALL cetObjectGetProperty(obj%, “AbsPos”, ADDROF(pos%))
MACRO call:
PrinterAbsPosition(obj%, pos%)

IsEmpty(Integer, RO) line buffer status, 1 – empty, 0 – not empty

CALL call:
CALL cetObjectGetProperty(obj%, “Empty”, ADDROF(empty%))
MACRO call:
PrinterIsEmpty(obj%, empty%)

WordWrap(Integer, WO) Turn word wrapping on(1) or off(0)

CALL call:
CALL cetObjectSetProperty(obj%, “WordWrap”, wrap%)
MACRO call:
PrinterWordWrap(obj%, wrap%)

Notes

1. Fonts are enumerated starting with 1 in order that they were created by **Font** call. Font number 0 is default printer font which is created when Create call is performed.
   It is recommended to create all fonts required immediately after Create call, then select fonts by its number.
2. Page sizes and position in NewPosition call are in device units (pixels).

Example BASIC program

Example programs prt.b and fax.b are included into setup.exe and installed into cetsamp\CBExt directory.
Prt.b shows basic calls in normal mode.
Fax.b composes simple fax message using virtual mode and drawing calls.