PostScript®
UNICODE EXTENSION REFERENCE

davidnewall.com

PostScript Extensions

This document contains detailed information about extensions to the PostScript language to support UNICODE and UTF-8-encoded strings. It is divided into two sections:

- Section 1 gives a summary of the extensions.
- Section 2 provides detailed descriptions of all extensions.

Each description is presented in the same format as the PostScript Language Reference:

extension *argument*$_1$ ... *argument*$_n$ **extension** *result*$_1$ ... *result*$_m$

A detailed explanation of the extension.

Example

An example of the use of this extension.

The symbol ⇒ designates the values left on the operand stack by the example.

Errors: A list of errors that this extension might execute
See Also: A list of related extensions

At the head of an extension description, *argument*$_1$ through *argument*$_n$ are the arguments that the extension requires, with *argument*$_n$ being the topmost element on the operand stack. The extension pops these objects from the operand stack and consumes them; then it executes. After executing, the extension leaves the objects *result*$_1$ through *result*$_m$ *on the stack, with result*$_m$ *being the topmost element.*

Normally, the operand and result names suggest either their types or their uses. Table 1 explains some commonly used names (other than basic type names).

UNICODE Extension Summary

string	**utf8decode** *array*		decode UTF-8 string giving array of UNICODE values
dict array	**unicodeshow**	-	paint glyphs for *array* in current font
a_x a_y *dict array*	**aunicodeshow**	-	Add (a_x, a_y) *to width of each glyph while showing array*
c_x c_y *int dict array*	**widthunicodeshow**	-	Add (c_x, c_y) *to width of glyph for int* while showing *array*
int a_x a_y c_x c_y *dict array*	**awidthunicodeshow**	-	Combine effects of *aunicodeshow* and *widthunicodeshow*
dict array numarray|numstring	**xunicodeshow**	-	Paint glyphs for *array* using *x* widths in *numarray|numstring*
dict array numarray|numstring	**xyunicodeshow**	-	Paint glyphs for *array* using *x* and *y* widths in *numarray|numstring*
dict array numarray|numstring	**yunicodeshow**	-	Paint glyphs for *array* using *y* widths in *numarray|numstring*
dict array	**unicodestringwidth**	w_x w_y	Return width of glyphs for *array* in current font
proc array	**kunicodeshow**	-	Execute *proc* between glyphs shown from *array*

UNICODE Extension Details

aunicodeshow *a_x a_y dict array* **aunicodeshow** -

Paints glyphs for the UNICODE values in *array* in a manner similar to **unicodeshow**; however, while doing so, **aunicodeshow** adjusts the width of each glyph shown by adding *a_x* to the glyph's *x* width and *a_y to its y* width, thus modifying the spacing between glyphs. The numbers *a_x and a_y are x* and *y* displacements in the user coordinate system, not in the glyph coordinate system.

This extension enables fitting a string of text to a specific width by adjusting all the spacing between glyphs by a uniform amount. For a discussion of glyph widths, see Section 5.4, "Glyph Metric Information" of *PostScript Language Reference.*

Example

```
AdobeGlyphList dup length dict begin {
  exch [ exch currentdict 3 index known
    {currentdict 3 index get aload pop}
  if ] def
} forall currentdict end /Map exch def
/Helvetica 8 selectfont
14 67 moveto
"NormalSpace"   Map [8220 78 111 114 109 97 108 83 112 97 99 101 8221] unicodeshow
14 47 moveto 4 0
"WideSpace" Map [8220 87 105 100 101 83 112 97 99 101 8221] aunicodeshow
```

Errors: **invalidfont, nocurrentpoint, typecheck**
See Also: **aunicodewidthshow, kunicodeshow, unicodeshow widthunicodeshow, xunicodeshow, xyunicodeshow, yunicodeshow**

awidthunicodeshow *c_x c_y int a_x a_y dict array* **awidthunicodeshow** -

Paints glyphs for the UNICODE values in *array* in a manner similar to **unicodeshow**, but combines the special effects of **aunicodeshow** and **widthunicodeshow**. **awidthunicodeshow** adjusts the width of each glyph shown by adding *a_x to its x* width and *a_y to its y* width, thus modifying the spacing between glyphs. Furthermore, **awidthshow** modifies the width of each occurrence of the glyph for the UNICODE value *int* by an additional amount (c_x, c_y).

This extension enables fitting a string of text to a specific width by adjusting the spacing between all glyphs by a uniform amount, while independently controlling the width of the glyph for a specific character, such as the space. For a discussion of glyph widths, see Section 5.4,"Glyph Metric Information" of *PostScript Language Reference.*

Example

```
AdobeGlyphList dup length dict begin {
  exch [ exch currentdict 3 index known
    {currentdict 3 index get aload pop}
  if ] def
} forall currentdict end /Map exch def
/Helvetica 8 selectfont
14 67 moveto
"Normal Space"  Map [8220 78 111 114 109 97 108 32 83 112 97 99 101 8221] unicodeshow
14 47 moveto
5 0 16#20 2 0
"Wide Space"  Map [8220 87 105 100 101 32 83 112 97 99 101 8221] awidthunicodeshow
```

Errors: **invalidfont, nocurrentpoint, typecheck**
See Also: **aunicodeshow, kunicodeshow, unicodeshow, widthunicodeshow xunicodeshow, xyunicodeshow, yunicodeshow**

kunicodeshow *proc dict array* **kunicodeshow** -

Paints glyphs for the UNICODE values in *array* in a manner similar to **unicodeshow**, but allows program intervention between characters. If the values in *array* are int_0, int_1, ... int_n, **kunicodeshow** proceeds as follows: First it shows the glyph for int_0 at the current point, updating the current point by the width of that glyph. Then it pushes int_0 and int_1 on the operand stack and executes *proc*. *proc* may perform any actions it wishes; typically, it will modify the current point to affect the subsequent placement of the glyph for int_1. **kunicodeshow** continues by showing the glyph for int_1, pushing int_1 and int_2 on the stack, executing *proc*, and so on. It finishes by pushing int_{n-1} and int_n on the stack, executing *proc*, and finally showing the glyph for int_n.

When proc is called for the first time, the graphics state (in particular, the current transformation matrix) is the same as it was at the time **kunicodeshow** was invoked, except that the current point has been updated by the width of the glyph for int_0. Execution of *proc* is permitted to have any side effects, including changes to the graphics state. Such changes persist from one call of *proc* to the next and may affect graphical output for the remainder of **kunicodeshow**'s execution and afterward. When *proc* completes execution, the value of currentfont is restored. The name **kunicodeshow** is derived from "kern-show." To kern glyphs is to adjust the spacing between adjacent glyphs in order to achieve a visually pleasing result. The **kunicodeshow** operator enables user-defined kerning and other manipulations, because arbitrary computations can be performed between pairs of glyphs.

kunicodeshow can be applied only to base fonts. If the current font is a composite font or a CIDFont, or defines neither CharProcs not CharStrings, an **invalidfont** error occurs.

Errors: **invalidfont, nocurrentpoint, typecheck**
See Also: **aunicodeshow, awidthunicodeshow, kunicodeshow, unicodeshow, widthunicodeshow, xunicodeshow, xyunicodeshow, yunicodeshow**

unicodeshow *dict array* **unicodeshow** -

Paints glyphs for the UNICODE values in *array* on the current page starting at the current point, using the font face, size, and orientation specified by the current font (as returned by **currentfont**).

dict maps UNICODE values to potential glyph names, either as a single name or an array of names. The current font's **CharStrings** dictionary is searched for each potential glyph name, and the first glyph found is selected. If the current font has no *CharStrings* dictionary, it's **CharProcs** dictionary is used instead. An **invalidfont** error occurs if the font has neither **CharStrings** nor **CharProcs**.

If a UNICODE value is not mapped by *dict*, or if none of the potential glyph names appears in the font's **CharStrings** dictionary, **uniXXXX** is used as a fallback name for values less than 16#10000 and **uXXXXXXXX** for values of 16#10000 and over, where XXXXXX is the value in hexadecimal.

The spacing from each glyph to the next is determined by the glyph's width, which is an (*x, y*) displacement that is part of the glyph description. When it is finished, **unicodeshow** adjusts the current point in the graphics state by the sum of the widths of all the glyphs shown. **unicodeshow** requires that the current point initially be defined (for example, by **moveto**); otherwise, a **nocurrentpoint** error occurs.

See Chapter 5 of *PostScript Language Reference* for complete information about the definition, manipulation, and rendition of fonts.

Errors: **invalidfont, nocurrentpoint, typecheck**
See Also: **moveto, setfont, aunicodeshow, aunicodewidthshow, kunicodeshow widthunicodeshow, xunicodeshow, xyunicodeshow, yunicodeshow**

unicodestringwidth *dict array* **unicodestringwidth** w_x w_y

Calculates the change in the current point that would occur if *dict* and *array* were given as operands to **unicodeshow** with the current font. w_x and w_y are computed by adding together the width vectors of all the individual glyphs for *array* and converting the result to user space. They form a distance vector in the x and y dimensions describing the width of the glyphs for the entire string in user space. See Section 5.4, "Glyph Metric Information," of *PostScript Language Reference* for a discussion of glyph widths.

To obtain the glyph widths, **unicodestringwidth** executes the descriptions of one or more of the glyphs in the current font and may cause the results to be placed in the font cache. However, **unicodestringwidth** prevents the graphics operators that are executed from painting anything onto the current page.

Note that the width returned by **unicodestringwidth** is defined as movement of the current point. It has nothing to do with the dimensions of the glyph outlines (see **charpath** and **pathbbox**).

Errors: **invalidfont, typecheck**
See Also: **setfont, unicodeshow**

utf8decode *string* **utf8decode** *array*

Returns an array object containing UNICODE values decoded from *string*, which is encoded using UTF-8 variable-width character encoding. Invalid UTF-8 sequences are replaced with Unicode Character 'REPLACEMENT CHARACTER' (U+FFFD).

Example

```
(\342\200\234UTF-8\342\200\235) utf8decode  ⇒  [8220 85 84 70 45 56 8221]
```

widthunicodeshow c_x c_y *int dict array* **widthunicodeshow** -

Paints glyphs for the UNICODE values in *array* in a manner similar to **unicodeshow**; however, while doing so, it adjusts the width of each occurrence of the glyph for UNICODE value *int* shown by adding c_x to its *x* width and c_y *to its y* width, thus modifying the spacing between it and the next glyph. This operator enables fitting a string of text to a specific width by adjusting the width of the glyph for a specific character, such as the space character.

Example

```
AdobeGlyphList dup length dict begin {
  exch [ exch currentdict 3 index known
    {currentdict 3 index get aload pop}
  if ] def
} forall currentdict end /Map exch def
/Helvetica 8 selectfont
14 67 moveto
"Normal Space"  Map [8220 78 111 114 109 97 108 32 83 112 97 99 101 8221] unicodeshow
14 47 moveto 6 0 16#20
"Wide  Space"   Map [8220 87 105 100 101 32 83 112 97 99 101 8221] widthunicodeshow
```

Errors: **invalidfont, nocurrentpoint, typecheck**
See Also: **aunicodeshow, aunicodewidthshow, kunicodeshow, unicodeshow xunicodeshow, xyunicodeshow, yunicodeshow, unicodestringwidth**

xunicodeshow *dict array numarray* **xunicodeshow** -
dict array numstring **xunicodeshow** -

Is similar to **xyshow**; however, for each glyph shown, *xshow* extracts only one number from *numarray* or *numstring*. It uses that number as the *x* displacement and the value 0 as the y displacement. In all other respects, **xshow** behaves the same as **xyshow**.

Errors: **invalidfont, nocurrentpoint, typecheck**
See Also: **unicodeshow, xyunicodeshow, yunicodeshow**

xyunicodeshow *dict array numarray* **xyunicodeshow** -
dict array numstring **xyunicodeshow** -

Paints glyphs for the UNICODE values in *array* in a manner similar to **unicodeshow**. After painting each glyph, it extracts two successive numbers from the array *numarray* or the encoded number string *numstring*. These two numbers, interpreted in user space, determine the position of the origin of the next glyph relative to the origin of the glyph just shown. The first number is the *x* displacement and the second number is the *y* displacement. In other words, the two numbers override the glyph's normal width.

If *numarray* or *numstring* is exhausted before all values in *array* have been shown, a **rangecheck** error occurs. See Section 5.1.4, "Glyph Positioning," in *PostScript Language Reference* for information about **xyshow**, and Section 3.14.5, "Encoded Number Strings," for an explanation of the *numstring* operand.

Errors: **invalidfont, nocurrentpoint, rangecheck, typecheck**
See Also: **unicodeshow, xunicodeshow, yunicodeshow**

yunicodeshow *dict array numarray* **yunicodeshow** -
dict array numstring **yunicodeshow** -

Is similar to **xyshow**; however, for each glyph shown, *yshow* extracts only one number from *numarray* or *numstring*. It uses that number as the *y* displacement and the value 0 as the x displacement. In all other respects, **yshow** behaves the same as **xshow**.

Errors: **invalidfont, nocurrentpoint, rangecheck, typecheck**
See Also: **unicodeshow, xyunicodeshow, yunicodeshow**