Adobe Enterprise & Developer Support Adobe Technical Note #5902

# Generating PostScript Names for Fonts Using OpenType Font Variations

Version 1.0, September 14 2016

## 1 Introduction

This document specifies how to generate PostScript names for OpenType variable font instances. Please see the OpenType specification chapter, OpenType Font Variations Overview for an overview of variable fonts and related terms, and the chapter fvar - Font Variations Table for the terms specific to this discussion.

The 'fvar' table optionally allows for PostScript names for named instances of an OpenType variations fonts to be specified. However, the PostScript names for named instances may be omitted, and there is no mechanism to provide a PostScript name for instances at arbitrary points in the variable font design space. This document describes how to generate PostScript names for such cases. This is useful in several workflows. For the foreseeable future, arbitrary instances of a variable font must be exported as a non-variable font for legacy applications and for printing: a PostScript name is required for this.

The primary goal of the document is to provide a standard method for providing human readable PostScript names, using the instance font design space coordinates and axis tags. A secondary goal is to allow the PostScript name for a variable font instance to be used as a font reference, such that the design space coordinates of the instance can be recovered from the PostScript name. However, a descriptive PostScript name is possible only for a limited number of design axes, and some fonts may exceed this. For such fonts, a last resort method is described which serves the purpose of generating a PostScript name, but without semantic content.

## 2. PostScript Name Generation Algorithm

First, create a family prefix for the PostScript name.

1. If the font provides a Variations PostScript Name Prefix string (name ID 25), use that.

2. Otherwise create the family prefix as follows:
    a. Start with the US English string for typographic family name (name ID 16).
    b. Remove any characters other than ASCII-range uppercase Latin letters, lowercase Latin letters, and digits.

To this family prefix, append the style information as follows.

For named font instances:

1. Start with "-" (U+002D).

Append the US English string for the instance's subfamilyNameID in the 'fvar' table.

2. Except for the initial "-" (U+002D), remove any characters other than ASCII-range uppercase Latin letters, lowercase Latin letters, and digits.

For arbitrary font instances:

For each variation axis, create and append an axis value descriptor as follows. The axis value descriptors do not need to be in the same order as the axis entries in the 'fvar' table.

1. Start with "_" (U+005F).
2. Append the decimal representation of the (16.16 fixed) user space style coordinate for the axis, using the minimum number of decimal places as are needed such that round-tripping from fixed to decimal and back to fixed will not lose any precision (see appendix below for sample Python code). Omit all leading and trailing zeroes, and if there is no fractional part, omit the decimal point as well. Represent a coordinate value of zero simply as "0" (U+0030). Do not precede non-negative values by "+" (U+002B).
3. Append the variation axis tag from the 'fvar' table (e.g., "wght"), removing any trailing spaces (0x20) in the tag. An axis value descriptor may be entirely omitted if its coordinate value is the default one for the axis; doing so is especially recommended when the PostScript name length approaches the 127 character limit.

Named instances may also use this method, even though it may result in names that are less human-readable.

If after constructing the PostScript name in this way the length is greater than 127 characters, then construct the "last resort" PostScript name as follows:

1. Start with the family prefix, as constructed above.
2. Append "-" (U+002D).
3. Append some other identifier for this instance, using only ASCII-range digits and uppercase and lowercase Latin letters, such that the implementation or workflow in which the PostScript name is used could have the option of using the entire PostScript name as a key to some external representation for the style information.
4. Append "...", three period characters (U+002E).

Including a Variations PostScript Name Prefix string (name ID 25) in a font could be useful in the following cases:

5. if the US English typographic family name, US English named instance fvar subfamilyNameID, or the number of axis descriptors in the font could tip the length of the generated PostScript names to over 127 characters, or
6. if the US English typographic family name contains accented or other characters that when removed by the algorithm above could cause confusion or even ambiguity in PostScript names. For example, both typographic family names “André Sans” and “Andró Sans” resolve to family prefix “AndrSans”, an ambiguity that could be avoided by providing Variation PostScript Name Prefixes “AndreSans” and “AndroSans” in the fonts.

Examples:

| **US English name ID 16** | **Variations PostScript Name Prefix string** | **Instance info** | **Generated PostScript name** |
|---|---|---|---|
| **“Andre Var”** | (absent) | Named instance “Black” | “AndreVar-Black” |
| **“Andre-Var!”** | (absent) | Named instance “Extra-Bold” | “AndreVar-ExtraBold” |
| **“Andre Var”** | (absent) | Arbitrary instance | “AndreVar_900wght_5.5wdth” |
| **“Andre Var”** | (absent) | Arbitrary instance | “AndreVar_-2.9wght_-1.4wdth” |
| **“André Var”** | (absent) | Named instance “Black” | “AndrVar-Black” |
| **“Andró Var”** | (absent) | Named instance “Black” | “AndrVar-Black” |
| **“André Var”** | “AndreVar” | Named instance “Black” | “AndreVar-Black” |
| **Some very long name** | “FamNm” | Many variation axes | “FamNm-” followed by axis |
| **“Andre Var”** | “AndreVar” | Many variation axes (too many to fit into 127 chars), last resort | “AndreVar-48AB868...” |

Appendix: Sample Code for Value Conversion

The following Python code implements the precision requirements of the “axis descriptors” method for generating PostScript names. Set the precisionBits parameter to 16. The code is taken from the fixedTools.py file in the github fonttools Python package (as of 22 July 2016):

```
def fixedToFloat(value, precisionBits):
     """Converts a fixed-point number to a float, choosing the float
     that has the shortest decimal representation.  To convert a
```

```
      fixed number in a 2.14 format, use precisionBits=14.  This is
      pretty slow compared to a simple division.  Use sporadically.
precisionBits is only supported up to 16.
      """
      if not value: return 0.0

      scale = 1 << precisionBits
      value /= scale
      eps = .5 / scale
      lo = value - eps
      hi = value + eps
      # If the range of valid choices spans an integer, return the
integer.
      if int(lo) != int(hi):
            return float(round(value))
      fmt = "%.8f"
      lo = fmt % lo
      hi = fmt % hi
      assert len(lo) == len(hi) and lo != hi
      for i in range(len(lo)):
            if lo[i] != hi[i]:
                  break
      period = lo.find('.')
      assert period < i
      fmt = "%%.%df" % (i - period)
      value = fmt % value
      return float(value)

def floatToFixed(value, precisionBits):
      """Converts a float to a fixed-point number given the number of
      precisionBits.
      """
      return int(round(value * (1<<precisionBits)))
```