Special-Purpose OpenType Japanese Font Tutorial: Kazuraki

Technical Note #5901

ADOBE SYSTEMS INCORPORATED
Corporate Headquarters
345 Park Avenue
San Jose, CA 95110-2704
(408) 536-6000

May 31, 2007

Contents

Introduction . . . 1

Kazuraki Design Motivations . . . 1

- Genuine Proportional Glyphs . . . 1
- Vertical-Only Hiragana Ligatures . . . 2
- CID- Versus Name-Keyed Structure . . . 2

OpenType Table Settings & Overrides . . . 2

- BASE . . . 2
- CFF . . . 3
- GPOS . . . 5
- GSUB . . . 5
- OS/2 . . . 6
- VORG . . . 6
- cmap . . . 7
- name . . . 7
- vmtx . . . 7

Special Tools . . . 8

OpenType Control Files & Data . . . 8

- CIDFont File . . . 8
- The “features” File . . . 8
- The “FontMenuNameDB” File . . . 9
- CMap Files . . . 9

Testing & Compatibility Considerations . . . 9

1 Special-Purpose OpenType Japanese Font Tutorial: Kazuraki

1.1 Introduction

This tutorial is designed to guide Japanese font developers in building special-purpose OpenType® Japanese fonts, using KazurakiStd-Light (to be referred to as simply Kazuraki® from this point forward) as an example of how to build a truly proportional Japanese font. The techniques, tools, and control files that are described or referenced in, or attached to, this document are tightly coupled to tools that are included in AFDKO (Adobe® Font Development Kit for OpenType) Version 2.0 or greater, which is available, at no charge, at the following URL:

http://partners.adobe.com/public/developer/opentype/afdko/topic.html

Please pay close attention to the last section of this document, which includes information—relevant as of this writing—about compatibility with applications.

If you have any questions regarding the content of this document, please do not hesitate to contact its author, Ken Lunde (*lunde@adobe.com*).

1.2 Kazuraki Design Motivations

With the exception of the vertical-only hiragana ligatures, all glyphs in Kazuraki have corresponding horizontal and vertical forms. That is, for each character, there are two glyphs in the font. The glyphs themselves are the same, but their set widths and default positioning along both X- and Y-axes are different. All glyphs needed to be replicated for vertical use, due to limitations in the ability to shift glyphs in both X- and Y-axis directions in the OpenType 'vmtx' table, coupled with the strong desire to make expected behavior the default—without an application depending on, or activating, any GSUB or GPOS features.

The glyph complement includes glyphs for all standard kana characters, but includes only a small set of kanji, 297 to be exact, suitable for creating Japanese greeting cards. Kazuraki also includes a basic set of proportional Latin glyphs to aid in keyboard input. At a minimum, we recommend including glyphs for ASCII (U+0020 through U+007E). While Kazuraki is fully-functional in this limited context, it also serves as an example for building comparable fonts.

1.2.1 Genuine Proportional Glyphs

All horizontal/vertical glyph pairs in Kazuraki are the same, with the exception of small kana, some punctuation, and parenthetical symbols, which require different glyphs for horizontal and vertical use in conventional Japanese fonts, due to their position or orientation. Post-processing of the glyph data is used to derive the vertical versions, which are positioned along the Y-axis differently, and have different metrics.

1.2.2 Vertical-Only Hiragana Ligatures

Kazuraki includes a small number of vertical-only hiragana ligatures, seventeen to be exact. Four are three-character ligatures, and the remaining thirteen are two-character ones. These are activated through the use of the 'liga' GSUB feature. The advantage in using 'liga' is that most applications that perform an adequate level of typesetting also tend to automatically invoke this GSUB feature, or at least allow users to activate it through a standard UI. This allows vertical-only hiragana ligatures to be used by default in many applications.

1.2.3 CID- Versus Name-Keyed Structure

Kazuraki was built as a CID-keyed OpenType font. Its 'CFF' table is built from a CIDFont file. Although Kazuraki could have been built as a name-keyed font, CID-keyed fonts have advantages for Japanese fonts. The primary advantage is that a CID-keyed structure supports multiple hint dictionaries. Each hint dictionary ideally covers glyphs for specific glyph classes, and each hint dictionary can have its own hinting parameters. Using multiple hint dictionaries thus offers significant rendering advantages.

1.3 OpenType Table Settings & Overrides

Many of the OpenType tables require special settings for Kazuraki. This section describes the special settings, one table at a time, along with information that demonstrates how the table overrides are specified in the "features" file as used as input for AFDKO's *makeotf* tool.

1.3.1 BASE

There is no special treatment necessary for the 'BASE' table, other than the usual ICF (Ideographic Character Face) and baseline values that should be normally specified. It is very important to include a 'BASE' table in all OpenType fonts. The following is the 'BASE' table override in the "features" file of Kazuraki:

```
table BASE {
  HorizAxis.BaseTagList                 icfb  icft  ideo  romn;
  HorizAxis.BaseScriptList  DFLT  ideo   -68  828   -120  0,
                            hani  ideo   -68  828   -120  0,
                            kana  ideo   -68  828   -120  0,
                            latn  ideo   -68  828   -120  0;

  VertAxis.BaseTagList                  icfb  icft  ideo  romn;
  VertAxis.BaseScriptList   DFLT  ideo  52    948   0     120,
                            hani  ideo  52    948   0     120,
                            kana  ideo  52    948   0     120,
                            latn  ideo  52    948   0     120;
} BASE;
```

Note that only the ‘DFLT’, ‘hani’, ‘kana’, and ‘latn’ scripts are declared in the ‘BASE’ table override, based on the glyph complement of Kazuraki.

1.3.2 CFF

The original source data for Kazuraki is a name-keyed OpenType font containing exactly 542 glyphs, each with 1000-unit set widths. The same source font also contains ‘palt’ and ‘vpal’ GPOS features that specify the desired glyph metrics (horizontal and vertical set widths, and X- and Y-axis shifting values, respectively), and these GPOS features are used as the source of the default metrics for the horizontal and vertical glyphs in the final font, which is an Adobe-Identity-0 CID-keyed OpenType font containing 1,223 glyphs (CIDs 0 through 1222).

Three AFDKO tools are used to process the original set of 542 glyphs: *tx*, *mergeFonts* and *rotateFont*.

The first tool, *tx*, simply extracts the ‘CFF’ table from the source OpenType font, and converts it into a name-keyed Type 1 font, using the following command line:

```
% tx -t1 Kazuraki_2.5.otf > font.ps
```

The second tool, *mergeFonts*, is used to convert the glyph names into CIDs, and to simultaneously synthesize the vertical glyphs from the original horizontal versions, using the following command line:

```
% mergeFonts -cid cidfontinfo cidfont.raw h.map font.ps v.map font.ps
```

The end result is an Adobe-Identity-0 CIDFont file, named “cidfont.raw,” containing 1,008 glyphs in a single hint dictionary.

The third tool, *rotateFont*, serves to set the widths for the horizontal glyphs, and to also shift them along the X-axis. The set width and X-axis shifting values are in the ‘palt’ GPOS feature of the source name-keyed OpenType font. The following command line is used:

```
% rotateFont -t1 -rtf shift.map cidfont.raw cidfont-prop.raw
```

The end result is an Adobe-Identity-0 CIDFont file, named “cidfont-prop.raw,” containing 1,008 glyphs, the horizontal versions of which now have proportional metrics. 215 proportional Latin glyphs are added to bring the glyph complement up to 1,223 glyphs.

As an example, let us explore the treatment of the horizontal glyph for the hiragana character “shi” (し). The glyph in the original name-keyed font is named “CID864” (named after CID=864 of the Adobe-Japan1-*x* character collection). Its ‘palt’ GPOS feature settings were as follows in the source name-keyed OpenType font:

```
position \CID864 <-223 0 -485 0>;
```

After processing by the *mergeFonts* tool, this (horizontal) glyph becomes CID=45. The ‘palt’ data shown above is used to generate the following *rotateFont* directive for the “shift.map” mapping file:

```
45 45 515 -223 0
```

The calculation is simple: the "-223" is used as-is as the X-axis shifting value, and the default set-width value of 1000 becomes 515 after the "-485" is added to it (a subtraction operation).

The set widths and Y-axis shifting for the vertical glyphs are specified in a 'vmtx' table override definition that is inserted into the "features" file. The handling of vertical glyphs, in terms of specifying their set widths and Y-axis positions, is covered later in this document.

Once these tools have been run, and the glyphs are assigned to CIDs, and the horizontal glyphs have been set to their default set widths and X-axis positions (that is, made proportional), the CIDFont is then hinted as usual. The process of hinting also involves creating multiple hint dictionaries, ideally only one for each glyph class.

NOTE: The process of establishing multiple hint dictionaries in a CIDFont requires files and tools that are not included in AFDKO, and their description is intentionally (and appropriately) omitted from this document. However, *mergeFonts* techniques described in Adobe Tech Note #5900 ("AFDKO Version 2.0 Tutorial: mergeFonts, rotateFont & autohint"), which is among the documentation bundled with AFDKO, can be used to establish multiple hint dictionaries. Multiple *mergeFonts* mapping files, in which the first line each each file names a hint dictionary, is the appropriate technique. And, multiple *mergeFonts* mapping files can serve to specify glyphs for single hint dictionaries. In fact, the proprietary tool that was used to establish multiple hint dictionaries for Kazuraki uses *mergeFonts* to perform this task.

The Special-Purpose Adobe-Identity-0 Character Collection

Because the glyph complement of Kazuraki does not adhere to the Adobe-Japan1-6 character collection, and because it makes little sense to extend Adobe-Japan1-6 to accommodate such special-purpose fonts, the special-purpose Adobe-Identity-0 character collection is advertised in the CFF. Although "Adobe-Identity-0" does not explicitly specify that Kazuraki is a Japanese font, other table settings, along with proven heuristics, are used to make clear the fact that it is a Japanese font.

In essence, the advantage of using the Adobe-Identity-0 character collection is that there are no preconceived notions of language or script, making it possible to build CIDFonts based on dynamic glyph sets, much like TrueType and name-keyed OpenType fonts. The technique of using the Adobe-Identity-0 character collection should not be used to build general-purpose OpenType Japanese fonts. The Adobe-Japan1-*x* character collection should be used instead.

File Size Issues

Because the horizontal/vertical glyph pairs are identical, in terms of their outlines, the subroutinization ability of AFDKO's *makeotf* tool makes the resulting CFF table only slightly larger than that of the original source data, which contained roughly half the number of glyphs. The subroutinized 'CFF' table thus became approximately fifty percent the size of the unsubroutinized version.

Hinting Issues

Hinting, in terms of stem widths, is applied as usual for Kazuraki. Alignment zones, however, are another matter. The hint dictionaries for non-Latin glyph classes, such as kana and kanji, typically use the following BlueValues array:

```
/BlueValues [-250 -250 1100 1100] def
```

However, due to the larger (taller) than usual bounding boxes of the vertical-only hiragana ligatures, the hint dictionary for the kana glyphs require different values, in order to ensure that there are no alignment zones in contact with its glyphs. The "Kana" hint dictionary of Kazuraki uses the following BlueValues array:

```
/BlueValues [-1050 -1050 1800 1800] def
```

Furthermore, the "Kanji" hint dictionary of Kazuraki uses the following BlueValues array, due to the extent to which the shapes of its glyphs extend above and below the 1000×1000 em-box.:

```
/BlueValues [-650 -650 1200 1200] def
```

The following is the FontBBox for Kazuraki:

```
/FontBBox {-335 -1004 1474 1769} def
```

It was thus critical to select BlueValues values less than -1004 (the Y-axis low point) and greater than 1769 (the Y-axis high point), at least for the "Kana" hint dictionary.

1.3.3 GPOS

The only GPOS features that are included in Kazuraki are 'kern' and 'vkrn', for horizontal and vertical kerning, respectively. The 'palt' and 'vpal' GPOS features in the original source data served to drive the production process, to specify the horizontal/vertical set widths and X- and Y-axis shifting values. These GPOS features are not in the final form of the font, because they are not necessary. Their values were used to define the default glyph metrics.

1.3.4 GSUB

Kazuraki contains only two GSUB features: 'vert' and 'liga'. Although the conventional ordering of these features is 'liga' followed by 'vert', this font's vertical-only hiragana ligatures necessitates a different ordering, specifically 'vert' followed by 'liga'. As a general rule, the ordering of GPOS and GSUB features in the "features" file is important, because the same ordering is reflected in the 'GPOS' and 'GSUB' tables that are generated by AFDKO's *makeotf* tool.

The 'vert' GSUB feature substitutes the horizontal forms with their vertical versions. This feature covers the majority of the font. Once the 'vert' GSUB feature has been applied, the vertical-only hiragana ligatures can then be applied via the 'liga' GSUB feature.

OpenType-savvy applications that support vertical writing automatically invoke the ‘vert’ (or ‘vrt2’, if present) GSUB feature. These same applications also invoke the ‘liga’ GSUB feature by default, which then serves to activate (or make default) the vertical-only hiragana ligatures.

1.3.5 OS/2

Because the special-purpose Adobe-Identity-0 character collection is used for Kazuraki, several ‘OS/2’ table fields must be more carefully specified, such as the OS/2.unicodeRange and OS/2.codePageRange fields. For Kazuraki, these settings are specified in the “features” file as the following ‘OS/2’ table overrides:

```
XHeight 458;
CapHeight 698;
UnicodeRange 0 1 2 31 33 35 48 49 50 59 62 68;
CodePageRange 1252 932;
```

Note that the “XHeight” and “CapHeight” values are set to values that correspond to the proportional Latin glyphs that are in its glyph complement.

The “UnicodeRange” values correspond as follows:

0	Basic Latin
1	Latin-1 Supplement
2	Latin Extended-A
31	General Punctuation
33	Currency Symbols
35	Letterlike Symbols
48	CJK Symbols And Punctuation
49	Hiragana
50	Katakana
59	CJK Unified Ideographs
62	Alphabetic Presentation Forms
68	Halfwidth And Fullwidth Forms

The “CodePageRange” value of 1252 corresponds to “Latin 1,” and 932 corresponds to “JIS/Japan.”

These ‘OS/2’ table settings help to explicitly identify Kazuraki as a Japanese font.

1.3.6 VORG

The ‘VORG’ table is automatically generated when using AFDKO’s *makeotf* tool, and is derived from the settings and overrides of the ‘vmtx’ table. See the section for the ‘vmtx’ table for more information on ‘vmtx’ table settings and overrides.

1.3.7 cmap

The 'cmap' table for CID-keyed OpenType fonts is built using one or more CMap files. For Kazuraki, because it is based on the special-purpose Adobe-Identity-0 character collection, special-purpose CMap files are necessary. Because the vertical glyphs are accessible through the 'vert' GSUB feature, only the horizontal glyphs are mapped from Unicode code points.

1.3.8 name

The 'name' table is built as usual, setting English and Japanese strings, as appropriate. The only exception is the name.ID=20 string, which is not necessary due to the special-purpose nature of Kazuraki. The specification of 'name' table string is performed in the "FontMenuNameDB" and "features" files. Care must be taken to explicitly set Japanese as the script and language, for as many of the strings as possible, as appropriate.

For more information about specifying 'name' table strings for OpenType Japanese fonts, please refer to Adobe Tech Note #5149 ("OpenType-CID/CFF CJK Fonts: 'name' Table Tutorial"), available at the following URL:

http://partners.adobe.com/public/developer/en/font/5149.OTFname_Tutorial.pdf

1.3.9 vmtx

The 'vmtx' table plays an absolutely crucial role in building fonts such as Kazuraki, because it is in this table that the vertical set widths are specified, along with any Y-axis shifting. Anything specified in the 'vmtx' table becomes default behavior. Thus, OpenType-savvy applications that support vertical writing can use such fonts without modification.

Kazuraki's "features" file contains a very large number of "VertAdvanceY" and "VertOriginY" statements in its 'vmtx' table overrides. Nearly every vertical glyph required treatment by one or both of these 'vmtx' overrides.

As an example, let us explore the treatment of the vertical glyph for the hiragana character "shi" (し). The glyph in the original name-keyed font is named "CID864" (named after CID=864 of the Adobe-Japan1-*x* character collection). Its 'vpal' GPOS feature settings were as follows:

```
position \CID864 <0 -26 0 331>;
```

After processing by the *mergeFonts* tool, this (vertical) glyph became CID=538. The 'vpal' data shown above was used to generate the following 'vmtx' table overrides for the "features" file:

```
VertOriginY \538 906;
VertAdvanceY \538 1331;
```

The calculation is simple: the "-26" is subtracted from 880 (a fixed value that represents the top of the em-box) to become 906, which is the new origin, and the "331" is added to the default 1000-unit width to become 1331.

1.4 Special Tools

A single special-purpose tool was written, in Perl, to generate all of the control files and data in a single execution. The mapping files that controlled the execution of the *mergeFonts* and *rotateFont* tools were generated by this tool, as was the "features" files containing 'vmtx' table overrides and all GSUB and GPOS feature definitions. The raw data to build the Unicode (UTF-32) and Shift-JIS CMap files were also generated by this tool. Due to the large number of glyphs, and the complex relationships between them, it was important to create a tool to do this work, because doing so by hand would have been tedious, and also prone to error.

When writing a comparable tool, I found that it was very useful to maintain a mapping from the glyph names in the source font to the final CIDs. This made generating the raw data for the CMap file a much easier task. It also made other tasks easier.

1.5 OpenType Control Files & Data

Once the name- to CID-keyed conversion is complete, the usual control files and data, required by AFDKO's *makeotf* tool, must be generated or supplied. These control files and data are detailed in the following sections.

1.5.1 CIDFont File

Kazuraki's CIDFont file is constructed as usual, with an appropriate number of hint dictionaries, ideally one for each glyph class, and with Adobe-Identity-0 as its advertised ROS (Registry, Ordering, and Supplement, which are the three entries of the CIDSystemInfo dictionary). As stated earlier in this document, Kazuraki's CIDFont file contains 1,223 glyphs, specifically CIDs 0 through 1222. Kazuraki contains exactly six hint dictionaries, named as follows:

- KazurakiStd-Light-Alphabetic (20 glyphs)
- KazurakiStd-Light-Dingbats (22 glyphs)
- KazurakiStd-Light-Generic (one glyph)
- KazurakiStd-Light-Kana (371 glyphs)
- KazurakiStd-Light-Kanji (594 glyphs)
- KazurakiStd-Light-Proportional (215 glyphs)

1.5.2 The "features" File

The "features" file plays an important role, in that overrides to specific tables can be made, and GPOS and GSUB features can be defined. Kazuraki contains two GPOS features, 'kern' and 'vkrn', to specify horizontal and vertical kerning pairs, respectively. Two GSUB features,

'vert' and 'liga', are also included, whose relative order is important, as described earlier in this document. Lastly, the 'vmtx' table overrides, which are also used to build the 'VORG' table, serve to specify the default vertical metrics.

1.5.3 The "FontMenuNameDB" File

The English and Japanese menu names that are recorded in the 'name' table of an OpenType font are specified in the "FontMenuNameDB" file. Kazuraki's "FontMenuNameDB" entry is shown below:

```
[KazurakiStd-Light]
          f=3,1,0x411,\304b\3065\3089\304d Std
          s=3,1,0x411,L
          c=3,1,0x411,\304b\3065\3089\304d Std L
          f=1,1,11,\82\a9\82\c3\82\e7\82\ab Std
          s=1,1,11,L
          c=1,1,11,\82\a9\82\c3\82\e7\82\ab Std L
          f=Kazuraki Std
          s=L
          c=Kazuraki Std L
```

It is important to stress that English menu names must be set—in addition to the obvious Japanese menu names—in case such fonts are used in applications whose heuristics may cause a failure to properly use the Japanese menu names.

1.5.4 CMap Files

For special-purpose fonts such as Kazuraki, only a Unicode CMap file is necessary. Even if there are no mappings outside the BMP, a UTF-32 CMap file is still recommended as input to AFDKO's *makeotf* tool. For Kazuraki, the Unicode CMap file was named "UniKazurakiStd-UTF32-H" to make it tightly coupled with the font. This CMap file is used solely as input to AFDKO's *makeotf* tool, to build the Unicode 'cmap' subtables of the resulting OpenType font.

For more information about building CMap files, please refer to Adobe Tech Note #5099 ("Building CMap Files for CID-Keyed Fonts"), available at the following URL:

http://partners.adobe.com/public/developer/en/font/5099.CMapFiles.pdf

1.6 Testing & Compatibility Considerations

Kazuraki works as expected in Adobe InDesign® CS2. The horizontal and vertical metrics are respected, and proper vertical layout is supported, including the vertical-only hiragana ligatures.

Kazuraki works in Adobe Illustrator® CS2 and Adobe Photoshop® CS2 with some limitations, mainly that the vertical-only hiragana ligatures do not function, even if the 'liga' GSUB feature is turned on.

In addition, these and other applications may not display Kazuraki's name in Japanese in their font menus. Kazuraki's name may instead display in English, using the English-language menu name strings that are specified in the 'name' table.

Kazuraki works very well with CS3 applications, such as InDesign CS3, Illustrator CS3, and Photoshop CS3. In fact, we recommend that CS3 applications be used for Kazuraki and comparable fonts.

Due to its unique (and limited) glyph complement, Kazuraki is not recommended for use as a component in these applications' Composite Font functionality.

When developing special-purpose OpenType Japanese fonts, it is prudent to rigorously test the font with a variety of operating systems and applications, to include entire document authoring workflows.