AFDKO Version 2.0 Tutorial: mergeFonts, rotateFont & autohint

Technical Note #5900

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

February 21, 2007

Contents

1 AFDKO Version 2.0 Tutorial: mergeFonts, rotateFont & autohint

1.1 Introduction

AFDKO (Adobe Font Development Kit for Opentype) Version 2.0, which was released in the latter part of 2006, introduces three new and very powerful font development tools, specifically *mergeFonts*, *rotateFont*, and *autohint*. These three new tools, along with careful and clever text-processing techniques, can be used to build a fully-functional CIDFont rollup tool. Of course, these tools can also be used for a wide variety of other purposes.

The *mergeFonts* tool can be run on name- and CID-keyed fonts, and can perform the following tasks: rename glyphs, replace glyphs, and add glyphs. Its input can be multiple font files, but will always create a single font file as output. The *rotateFont* tool can also be run on name- and CID-keyed fonts, and while its primary functionality is to rotate and shift glyphs, it can also be used to rename glyphs, and to change their widths. The *autohint* tool generates Type 1 hints, based upon the hinting parameters—alignment zones and stem width values—that are specified in the input font's hint dictionary (or hint dictionaries, in the case of CID-keyed fonts), which are then added to the CharStrings. The *autohint* tool can also be run on name- or CID-keyed fonts.

This tutorial is intended for font developers who use AFDKO Version 2.0 or higher, and provides information for three of its tools that go beyond the usage and help pages that are included, when invoked with the "-u" and "-h" command-line options, respectively.

While this initial version is written in English, translations into Chinese, Japanese, and Korean may be forthcoming. Until those translations are available, please note that this English version was specifically written to be clear, concise, and easy to understand.

If you have any questions regarding the *mergeFonts*, *rotateFont*, or *autohint* tools, or believe that you encountered a bug or otherwise odd behavior when using one of them, please do not hesitate to contact the author, Ken Lunde (*lunde@adobe.com*).

NOTE: The *mergeFonts* and *rotateFont* tools are sensitive to the line endings used in their respective mapping files. For example, when executing these tools on Mac OS X using the *Terminal* application, the mapping files must have Unix line endings (LF; 0x0A), not Macintosh ones (CR; 0x0D).

NOTE: The AFDKO tools described in this document cannot operate on Macintosh resource-fork font files. Type 1 fonts, for example, must be flat (that is, data-fork) files, which can be easily generated by commercially-available tools such as Fontographer or FontLab. FontLab Studio 5 refers to such Type 1 fonts as "ASCII/Unix Type 1" fonts, and uses ".pfa" as the default filename extension.

1.2 Using mergeFonts

The *mergeFonts* tool is extremely powerful. It extracts sets of glyphs from one or more name- or CID-keyed source fonts, then merges all of the extracted glyphs into a single name- or CID-keyed output font.

1.2.1 Mapping Files

A mapping file is required by *mergeFonts* when the input font or fonts are to be subsetting in any way, or if glyphs are to be renamed. The format of the file is simple. The first line must be set as follows:

```
mergeFonts
```

The first line unambiguously identifies the file as a *mergeFonts* mapping file. When working with CID-keyed fonts, a hint dictionary name can be specified immediately after the "mergeFonts" keyword, as follows:

```
mergeFonts KozMinProVI-Heavy-Kanji
```

The above line has the effect of assigning to the specified hint dictionary the CIDs that follow on the subsequent lines of the mapping file. If that hint dictionary does not exist in the CIDFont, it is created.

The lines that follow the "mergeFonts" line simply specify glyph name or CID pairs, whereby the first member of the pair (first column) is the source (new) glyph name or CID, and the second member (second column) is the add/replace (original or old) glyph name or CID. If no glyph name change is necessary, both members (columns) shall be identical. The following is an example of a valid *mergeFonts* mapping file:

```
mergeFonts KozMinProVI-Heavy-Kanji
1125      c21
1126      c22
1127      c23
1128      c24
```

In the above example, the glyph names "c21" through "c24" in the input font are renamed to CIDs 1125 through 1128, respectively, and are assigned to the hint dictionary named "KozMinProVI-Heavy-Kanji." Note that CIDs can also be expressed as five-digit zero-padded integers, so "1125" and "01125" equally represent CID=1125.

When generating *mergeFonts* mapping files, be aware that a ".notdef" (".notdef" for name-keyed fonts, or CID=0 for CID-keyed fonts) must be included or otherwise specified in at least one of the mapping files.

1.2.2 Command Lines

The *mergeFonts* tool is typically run using options and multiple input files. The "-g" and "-gx" options require a list of one or more glyph names or CIDs to follow, and they apply only to the

"source" font, which is the first input font if more than one input font is specified on the command line. The subsequent input fonts are referred to as "merge" fonts. The "-g" option specifies which glyphs to copy from the source font to the output font, and the "-gx" option specifies which glyphs to exclude from the source font. These options are useful when replacing glyphs in a font, and depending on the number of glyphs that are being replaced, one of these options is more appropriate than the other. For example, if the number of glyphs to be replaced is relatively small, it is easier to use the "-gx" option, and if the number of glyphs to be replaced is large, meaning that the number of unchanged glyphs is small, it is easier to use the "-g" option.

When specifying glyphs by CIDs, the "-g" and "-gx" options can use CID ranges as their argument. In fact, CIDs and CID ranges can be specified, through the use of commas and hyphens. The following argument specifies CIDs 1, 3, and 50 through 100:

```
1,3,50-100
```

When generating a CIDFont as output, the "-cid" option, followed by the name of the "cidfontinfo" file, is necessary. All mapping files must specify CIDs in the first column. Invoke *mergeFonts* with the "-h" option to see the format of a valid "cidfontinfo" file, along with an example. The purpose of the "cidfontinfo" file is to provide ROS (Registry, Ordering, and Supplement) information, along with other minimal information for a CIDFont file header.

When a mapping file is necessary, it appears before its corresponding merge font on the command line. For example, the following command specifies an output font, a source font, and two merge fonts, each of which has its own mapping file:

```
% mergeFonts font.out font.src map1.merge font1.merge \
map2.merge font2.merge
```

NOTE: By default, all glyphs in a font are added if no mapping file is present or specified.

NOTE: If there is a name conflict, the first font that is encountered by *mergeFonts* wins, in terms of which glyph is included in the output font, whether it is the source font or one of the merge fonts.

NOTE: The input font files may be a mixture of name- and CID-keyed fonts, but they must all be of the same CharString type, specifically Type 1 or Type 2 (CFF).

1.2.3 Renaming Glyphs

In order to rename one or more glyphs in a font, all glyphs in the font must be present in the mapping file, and only those glyphs whose names are to be changed shall have different source names specified in the first column. The following mapping file example serves to change the glyph named "exclam" into "c21":

```
mergeFonts
c21       exclam
```

Converting glyph names to CIDs requires the use of mapping files, and the following is an example mapping file that converts the glyphs named “c21,” “c22,” “c23,” and “c24” into CIDs 1125 through 1128, respectively:

```
mergeFonts
1125      c21
1126      c22
01127     c23
01128     c24
```

Note that the two different notations for specifying CIDs, one of which consists of five zero-padded digits, are used in the above example. In actual usage, a single notation should be used for consistency, to minimize confusion, and to maximize human readability.

1.2.4 Replacing Glyphs

When replacing glyphs, two or more input fonts are necessarily involved. The font that contains the glyph that is to be replaced is referred to as the “source” font, and the font or fonts that contain the glyphs that act as replacements are referred to as “merge” fonts.

The “-gx” option, followed by a list of the glyph names or CIDs to be replaced, is necessary. The “-gx” option applies to the source font only. It simply instructs *mergeFonts* to copy all glyphs except for those specified. If the number of glyphs to be replaced is large, meaning that the number of glyphs that remain as-is is small, it may be simpler to use the “-g” option, which specifies which glyphs to copy from the source font to the output font. And, the mapping files for the merge fonts must specify the same glyph names or CIDs, as appropriate, otherwise the resulting output font will not include a glyph by that name or CID.

For example, if we were to replace the glyph “c21” in font.src with the glyph “c21” in font.merge, the following command line is used:

```
% mergeFonts -gx c21 font.out font.src map.merge font.merge
```

A new font called “font.out” is created, which is generated from the contents of the source font (“font.src”) excluding the glyph named “c21,” and the glyph named “c21” is added from the merge font (“font.merge”). The contents of the “map.merge” file are as follows:

```
mergeFonts
c21       c21
```

1.2.5 Adding New Glyphs

When adding new glyphs, the “-g” or “-gx” options are not necessary. And, if all of the glyphs in the merge font are to be added as-is, including no changes to glyph names, no mapping files are necessary. If a subset of the glyphs in the merge fonts are to be added, or if glyph names are to be changed, mapping files are necessary. Consider the following examples.

The merge font contains only the glyph “c80” (no mapping file required):

```
% mergeFonts font.out font.src font.merge
```

The merge font contains other glyphs in addition to the glyph “c80” (mapping file required):

```
% mergeFonts font.out font.src map.merge font.merge
```

The contents of the “map.merge” file are as follows:

```
mergeFonts
c80       c80
```

1.2.6 Building CIDFonts

The *mergeFonts* tool can be used to build valid CIDFont files, using multiple merge fonts. The merge fonts can be name-keyed. For example, multiple name-keyed fonts can serve as the merge fonts, and by using the “-cid” option, along with mapping files that convert glyph names into appropriate CIDs, CIDFonts containing thousands or tens of thousands of glyphs can be generated within seconds.

The following command line and mapping files demonstrate how easily CIDFonts can be built using the *mergeFonts* tool:

```
% mergeFonts -cid cidfontinfo cidfont.ps NOTDEF.map NOTDEF.pfa \
hiragana.map hiragana.pfa katakana.map katakana.pfa r30.map r30.pfa \
r31.map r31.pfa
```

Contents of “NOTDEF.map” mapping file (one glyph: CID=0):

```
mergeFonts KozMinProVI-Heavy-Generic
0         NOTDEF
```

Contents of “hiragana.map” mapping file (83 glyphs: CIDs 842 through 924):

```
mergeFonts KozMinProVI-Heavy-Kana
842       smalla
843       a
… (79 lines omitted)
923       wo
924       nn
```

Contents of “katakana.map” mapping file (87 glyphs: CIDs 660 and 925 through 1010):

```
mergeFonts KozMinProVI-Heavy-Kana
660       extender
925       smalla
926       a
… (82 lines omitted)
1009      smallka
1010      smallke
```

Contents of “r30.map” mapping file (94 glyphs: CIDs 1125 through 1218):

```
mergeFonts KozMinProVI-Heavy-Kanji
1125      c21
1126      c22
… (90 lines omitted)
1217      c7D
1218      c7E
```

Contents of “r31.map” mapping file (94 glyphs: CIDs 1219 through 1312):

```
mergeFonts KozMinProVI-Heavy-Kanji
1219      c21
1220      c22
… (90 lines omitted)
1311      c7D
1312      c7E
```

Running the above command line, using the specified mapping files and merge fonts, a CIDFont containing 359 glyphs is created. The CIDs covered include 0, 660, 842 through 1010, and 1125 through 1312. And, a total of three hint dictionaries are created, as follows:

```
KozMinProVI-Heavy-Generic
KozMinProVI-Heavy-Kana
KozMinProVI-Heavy-Kanji
```

Note how the hint dictionary name specified on the first line of the “r30.map” and “r31.map” mapping files are the same. Specifying the hint dictionary name allows one to carefully control the hint dictionary structure of the resulting CIDFont file. Also, if multiple merge fonts have mapping files whose first line specifies the same hint dictionary name, the hint dictionary of the resulting CIDFont inherits its values from the first merge font that it encounters with that hint dictionary name specified.

If no hint dictionary name is specified, the glyphs from each merge font are assigned to a unique hint dictionary, named according to the “FontName” of the merge font, which is both inaccurate and inefficient.

Great care must be taken to control the hint dictionary structure of CIDFont files, and the *mergeFonts* tool is the vehicle for applying such control.

1.3 Using rotateFont

Although not nearly as powerful as the *mergeFonts* tool, the *rotateFont* tool also operates on name- and CID-keyed fonts, and can be used to rename, rotate, and shift glyphs, and for changing glyph widths. Glyph shifting is performed along the X- and Y-axes. While rotation and X- and Y-axis shifting can be specified on the command line and applied globally, mapping files can be used to apply X- and Y-axis shifting, and changes to glyph names and widths.

The current Adobe character collections specify pre-rotated forms for non–full-width glyphs, to be used for vertical writing, and made accessible through the ‘vrt2’ GSUB feature. For Japanese fonts, *Adobe-Japan1-3* was the first Adobe character collection to include these pre-rotated forms. The *rotateFont* tool trivializes the process of generating these pre-rotated glyphs from their original unrotated forms.

1.3.1 Command Lines

The *rotateFont* tool shall always be invoked with an option to specify the format of the output font, specifically "-t1" for a Type 1 font, or "-cff" for a CFF font. In either case, the output font will match the input font as to whether it is name- or CID-keyed.

Unlike the *mergeFonts* tool, the *rotateFont* tool does not perform conversion between name- and CID-keyed fonts. Also, unlike the *mergeFonts* tool, the first font specified on the command line is the input font, and the second font is the output font. Also, depending on whether changes are to be applied by a mapping file, or applied globally, the "-rtf" or "-rt" options, respectively, shall be used.

The following is an example *rotateFont* command line that applies 90-degree clockwise rotation, along with X- and Y-axis shifting of 120 and 880 units, respectively, with all three values being arguments of the "-rt" command-line option:

```
% rotateFont -t1 -rt 90 120 880 font.in font.out
```

1.3.2 Mapping Files

Glyph rotation and X- and Y-axis shifting, when applied to all glyphs in a font, does not require a mapping file. When using a mapping file, only those glyphs in the mapping file will be in the output font. The name of the mapping file follows the "-rtf" option. Note that a ".notdef" glyph entry is required. In the case of CID-keyed fonts, this means that an entry for CID=0 must be present in the mapping file.

The *rotateFont* mapping file contains five columns per line, as follows:

- Input glyph name or CID
- Output glyph name or CID
- Glyph width—use "None" to pass widths as-is
- X-axis shift—set to "0" (zero) for no change; negative values shift glyph to left
- Y-axis shift—set to "0" (zero) for no change; negative values shift glyph down

If the glyph name is to remain as-is, the input and output glyph names should be the same. Below is an example of a *rotateFont* mapping file line that passes the glyph named "exclam" as-is:

```
exclam exclam None 0 0
```

NOTE: If a mapping file is specified, its X- and Y-axis shift values take precendence over those specified as arguments for the "-rt" command-line option. In other words, if you must use both options—"-rt" and "-rtf"—the X- and Y-axis shifting values should be specified in the mapping file, and not as the second and third arguments of the "-rt" option.

NOTE: The order of the fields for the input and output glyph name is in a reverse order when compared to their order in the *mergeFonts* mapping file. In other words, the order of

the input/output glyph names in the *mergeFonts* and *rotateFont* mapping files reflects the order of the input/output font files as specified in their respective command lines.

1.3.3 Renaming Glyphs

The *rotateFont* tool can easily rename glyphs by simply specifying the new glyph name or CID in the second column of the mapping file. The following *rotateFont* mapping file line changes the name of the glyph named "exclam" to "c21":

```
exclam c21 None 0 0
```

If no glyph change is necessary, the glyph names or CIDs in the first and second columns shall be identical.

1.3.4 Rotating Glyphs

Glyph rotation can be performed by *rotateFont* only through the use of the "-rt" option, which expects three space-separated integer values as its arguments, as follows:

- Rotation, specified in degrees clockwise—use "0" (zero) for no rotation; negative values can be used to specify counter-clockwise rotation
- X-axis shift—use "0" (zero) for no X-axis shift
- Y-axis shift—use "0" (zero) for no Y-axis shift

Below is a command line example for rotating glyphs 90 degrees clockwise, followed by shifting the glyphs 200 units along the X-axis, and 800 units along the Y-axis:

```
% rotateFont -t1 -rt 90 200 800 font.in font.out
```

The following command line results in the same output font, but specifies counter-clockwise rotation through the use of the appropriate negative value:

```
% rotateFont -t1 -rt -270 200 800 font.in font.out
```

Note that the rotation performed by *rotateFont* is applied at coordinate 0,0, which is the glyph origin. Due to different baselines for different glyph classes, such as Latin versus non-Latin, or CJK versus non-CJK, X- and Y-axis shifting is generally necessary to reposition the rotated glyph within the em-box.

1.3.5 Changing Glyph Widths

The third column of the mapping file is used to specify glyph widths. The following mapping file line changes the glyph width to 1000 units, regardless of what its original width value was:

```
exclam exclam 1000 0 0
```

If the width is to remain as-is (unchanged), "None" shall be used, as exemplified below:

```
exclam exclam None 0 0
```

1.3.6 Glyph Shifting

Global shifting of glyphs is best performed on the command line, using the "-rt" option. The second and third arguments of the "-rt" option specify the X- and Y-axis shifting values, respectively. If only specific glyphs require shifting, a mapping file shall be used, specifying the X- and Y-axis shift values in the fourth and fifth columns, respectively. The following example shifts the glyph named "exclam" -100 units along the X-axis (to the left), and 250 units along the Y-axis (upward):

```
exclam exclam None -100 250
```

To perform no shifting, for one or both axes, a "0" (zero) shall be used, such as shown below:

```
exclam exclam None 0 250
exclam exclam None -100 0
exclam exclam None 0 0
```

1.3.7 Working With CIDFonts

The *rotateFont* tool can use CID-keyed fonts as input. Instead of specifying glyph names, CIDs are used instead. The following *rotateFont* mapping file sets the width of the glyph for CID=1200 to 1500 units, and also shifts it 250 units along the X-axis, which effectively centers the glyph within its new 1500-unit width, assuming that its original width was 1000 units:

```
0 0 None 0 0
1200 1200 1500 250 0
```

Note that the required CID=0 entry is include in the mapping file.

1.4 CIDFont Generation Notes

While the *rotateFont* and *mergeFonts* tools can be applied to CIDFonts, and can also be used to create CIDFonts, the CIDFont header that is generated is not complete, and some amount of careful text processing is required. For example, the following *FontInfo* dictionary entries may need to be added or adjusted: *Notice*, *FullName*, *FamilyName*, *Weight*, and *FSType*. The *XUID* array should also be set. The *CIDFontName*, *CIDFontVersion*, and *CIDSystemInfo* dictionary are properly set by the *mergeFonts* tool, according to the "cidfontinfo" file that is supplied as input.

The current Adobe character collections specify CID ranges that represent pre-rotated forms of all glyphs that are not considered to be full-width. This typically includes proportional- and half-width Latin glyphs, but is not limited to those glyph classes. See the corresponding Adobe Tech Note for each Adobe character collection for more information on the CID ranges for pre-rotated glyphs: *http://partners.adobe.com/public/developer/font/*

The *rotateFont* tool is ideal for generating the pre-rotated glyphs from their unrotated (upright) forms. The resulting font, containing pre-rotated glyphs, can be used as one of the many merge fonts that serve as input to the *mergeFonts* tool for building a fully-populated CIDFont.

1.5 Using autohint

After the *mergeFonts* and *rotateFont* tools are used to generate a CIDFont, there is another very useful tool that can be used, available in AFDKO Version 2.0 or higher, called *autohint*. The *autohint* tool can be run on name- and CID-keyed fonts, and uses the hinting parameters—*BlueValues* array, *OtherBlues* array, *StemSnapH* array, *StemSnapV* array, *StdHW*, and *StdVW*—to apply hinting to the CharStrings. You are encouraged to explore its command-line options to determine which are the most appropriate to use.

1.5.1 Calculating Alignment Zones & Stem Width Values

According to rendering-engine and hinting specialists, no hinting is better than poor hinting, and similarly, poorly-calculated hinting parameters will result in poor hinting, even when using the *autohint* tool. In addition, the paths for CharStrings shall adhere to the Type 1 CharString specification. For example, the control points—used for hinting purposes, specifically to determine which stems are to be controlled via hints, based on the supplied hinting parameters—shall be set at the extremes.

You are strongly encouraged to take great care in calculating the alignment zones and stem width values. The *stemHist* tool, also included in AFDKO Version 2.0 or higher, is designed to help in calculating alignment zones and stem width values by generating histogram data. The former functionality is invoked using the "-a" command-line option.

1.6 Verifying Results

The *tx* tool, which is included in AFDKO Version 2.0, is very useful for generating a glyph synopsis very quickly, for verifying the results of running the *mergeFonts* and *rotateFont* tools. Simply use the "-pdf" option, and redirect STDOUT to a PDF file. In fact, you are strongly encouraged to spend a couple of days, or a full week, to explore these tools, to discover their full potential. You will no doubt discover additional uses for these font development tools. Experimenting with these tools, then checking their output using the *tx* tool, is quite gratifying.

AFDKO Version 2.0 チュートリアル : mergeFonts, rotateFont & autohint

1.1 はじめに

2006 年後半にリリースされた AFDKO (Adobe Font Development Kit for Opentype) Version 2.0 には、mergeFonts、rotateFont、そして autohint という、３つの新しい強力なフォント開発ツールが導入されています。これらの新しいツールは、テキスト処理の手法をうまく併用することで、十分な機能を備えた CIDFont ロール・アップ・ツールとして利用することが可能ですが、他の幅広い用途にも利用できます。

mergeFonts ツールは名前キー方式または CID キー方式のフォントに対して実行、グリフ名の変更、グリフの置換および追加が可能です。複数のフォントファイルを入力とすることができますが、出力は常に単一のファイルとなります。rotateFont ツールも名前キー方式または CID キー方式のフォントや CID キーフォントに対して実行可能です。その主な機能はグリフの回転とシフトですが、グリフ名の変更やグリフの字幅の変更にも使用することができます。autohint ツールは、ヒントのパラメータ（並び域およびステム幅の値）に基づいて、Type 1 のヒントを生成します。これらのパラメータは入力フォントのヒント辞書（CID フォントの場合は複数のヒント辞書）中で指定されており、CharStrings に追加されます。autohint ツールも名前キー方式および CID キー方式のフォントの両方に対して実行可能です。

このチュートリアルは、AFDKO Version 2.0 またはそれ以降を使用するフォント開発者を対象にしたもので、上述の 3 つのツールに対する、コマンドライン上の「-u」および「-h」オプションで表示される解説とヘルプのページの内容を超える情報を提供することを意図しています。

mergeFonts、rotateFont、そして autohint の各ツールに関する質問、または使用中にバグまたは不具合と思われる点がありましたら、筆者の Dr. Ken Lunde（ケン・ランディ）の下記のメールアドレスまでご連絡ください。(lunde@adobe.com)

注： mergeFonts および rotateFont は、各々のマッピングファイルの行末の改行コードに敏感に反応します。例えば、Mac OS X で、ターミナルアプリケーションを使ってこれらのツールを実行する場合、マッピングファイルには Macintosh の改行コード（CR; 0x0D）ではなく、Unix の改行コード（LF; 0x0A）が含まれていなければなりません。

注： 本書で解説されている AFDKO ツールは、Macintosh の resource-fork フォントファイル上では動作しません。例えば、Type 1 フォントはフラットファイル（data-fork を指す）である必要があります。フラットファイルは、Fontographer や FontLab などの市販ツールで簡単に生成することができます。FontLab Studio 5 では、そのような Type 1 フォントを「ASCII/Unix Type 1」フォントと呼んでいて、初期設定ファイル名の拡張子としては「.pfa」を使用しています。

1.2 mergeFonts の使用法

mergeFonts は極めて強力なツールで、単一または複数の名前キー方式のソースフォントか、CID キー方式のソースフォントから種々のグリフの集合を抽出し、それらのグリフを結合して、単一の名前キー方式または CID キー方式のフォントに出力します。

1.2.1 マッピングファイル

入力フォント（複数可）からその部分集合を抽出する場合、あるいはグリフ名を変更する場合には、mergeFonts にはマッピングファイルが必要となります。そのファイル形式は単純で、まず最初の行で以下のように記述します。

```
mergeFonts
```

これにより、このファイルが mergeFonts のマッピングファイルであることが明確に識別可能になります。CID キー方式のフォントを使用する場合は、下記の例のように、「mergeFonts」のキーワードの直後にヒント辞書名を指定することができます。

```
mergeFonts KozMinPr6N-Heavy-Kanji
```

上の行を実行すると、マッピングファイルの後続行に記述された CID 番号を、指定のヒント辞書に割当てます。また、ヒント辞書が CIDFont 中に存在しない場合は、それを作成します。

「mergeFonts」以降の各行には、グリフ名または CID 番号をペア（対）だけを指定します。最初のカラムには、ソース（新規）のグリフ名か CID 番号を、そして二つ目のカラムには、追加／置換する（元の、または古い）グリフ名または CID 番号を記述します。グリフ名の変更が不必要であれば 、左右ともに同じものを記述します。以下は、mergeFonts のマッピングファイルの有効な記述例です。

```
mergeFonts KozMinPr6N-Heavy-Kanji
1125     c21
1126     c22
1127     c23
1128     c24
```

上記の例では、入力フォントのグリフ名「c21」から「c24」までが、それぞれに対応する CID 番号の「1125」から「1128」へと変更され、「KozMinPr6N-Heavy-Kanji」というヒント辞書に割り当てられます。また、CID 番号は、0 を加えた 5 桁の整数値でも表すことができます。例えば、「1125」は「01125」と表現でき、ともに CID+1125 を意味します。

mergeFonts のマッピングファイルを生成する際に、必ず 1 つの「.notdef」（名前キー方式のフォントでは「.notdef」、CID キー方式の フォントでは CID+0）を含めるか、最低 1 つのマッピングファイル中で指定する必要があります。

1.2.2 コマンドライン

mergeFonts ツールは一般に、オプションと複数の入力ファイルを使って実行します。オプション「-g」と「-gx」には、その後に 1 つまたは複数のグリフ名または CID 番号のリストが続いていなければならず、それらは「ソース」フォント（複数の入力フォントをコマンドライン上で指定した場合は、最初の入力フォント）にのみ適用されます。2 番目以降の入力フォントは、「併合」フォントと呼ばれます 。オプション「-g」は、ソースフォントから出力フォントへコピーするグリフを指定し、オプション「-gx」は、ソースファイルから除外するグリフを指定します。これらのオプションは、フォントの中のグリフを置換する際に便利で、置換するグリフの数によって、どちらのオプションが有効か変わってきます。例えば、置換するグリフの数が少ない場合は、「-gx」を使うほうが便利です。逆に、置換するグリフの数が多い（すなわち、変更のないグリフの数が少ない）場合は、「-g」を使うほうが簡単です。

CID 番号でグリフを指定する際は、「-g」および「-gx」のオプションとも、CID の範囲指定を引数として使うことができます。その際、コンマ「,」とハイフン「-」を用いて、CID 番号と CID の範囲を併用することも可能です。以下の引数は、CID 番号 1、3、そして 50 から 100 までを指定したものです。

```
1,3,50-100
```

CIDFont を出力として生成する際は、オプション「-cid」が必要となり、「cidfontinfo」ファイルの名前の前に指定します。どのマッピングファイルも、最初のカラムで CID を指定しなければなりません。オプション「-h」を用いて mergeFonts を起動し、有効な「cidfontinfo」ファイルの形式と、例を確かめてください 。「cidfontinfo」ファイルの目的は、ROS（Registry、Ordering および Supplement）の情報と、CIDFont ファイルのヘッダにに対するその他最小限の情報を提供することにあります。

マッピングファイルが必要な時は、コマンドライン上で、対応する併合フォントの前に指定します。以下のコマンドは、出力フォント、ソースフォント、そして 2 つの併合フォントを指定した例で、各フォントがそれぞれマッピングファイルを持っています。

```
% mergeFonts font.out font.src map1.merge font1.merge map2.merge font2.merge
```

注： マッピングファイルが存在しない場合や、指定されていない場合には、初期設定により、フォントのグリフがすべて追加されます。

注： 重複した名前がある場合、出力フォントにどのグリフが含まれるかという点では、ソースフォントか、併合フォントのどれかであるかを問わず、mergeFonts が最初に処理するフォントが優先されます。

注： 入力フォントのファイルには、名前キー方式のフォントと CID キー方式のフォントとが混在していてもかまいませんが、それらのフォントは同一の CharString の型、すなわち、Type 1 または Type 2 (CFF) のどちらかである必要があります。

1.2.3 グリフ名の変更

フォント内の単一のグリフ、あるいは複数のグリフの名前を変更する場合、フォント内のすべてのグリフがマッピングファイルに存在する必要があり、グリフ名を変更するグリフだけが、最初のカラムで異なるソース名を指定することになります。以下は、「exclam」という名前のグリフを「c21」に変更する マッピングファイルの例を示しています。

```
mergeFonts
c21     exclam
```

グリフ名を CID に変換するときはマッピングファイルの使用が不可欠となります。以下は、グリフ名「c21」、「c22」、「c23」、「c24」を、それぞれ CID 番号 1125 から 1128 までに変換するマッピングファイルの例です。

```
mergeFonts
1125    c21
1126    c22
01127   c23
01128   c24
```

上の例では、CID 番号を指定するには 2 種類の異なる表記方法（そのうちの 1 つは 0 を加えた 5 桁の整数値）が用いられていますが、実際には、どちらかの表記方法に統一する方が、混乱を最小限に抑えて、読みやすくなるので好ましいでしょう。

1.2.4 グリフの置換

グリフを置換する際には、 2 つまたはそれ以上の入力フォントが必要となります。置換されるグリフを含むフォントは「ソース」フォントと呼び、それに取って代わるグリフを含むフォントは「併合」フォントと呼びます。

置換されるグリフ名または CID 番号のリストの前には、オプション「-gx」を記述する必要があります。オプション「-gx」はソースフォントのみに適用されます。このオプションは、指定されたグリフ以外のすべてのグリフをコピーするよう mergeFonts に指示します。その際、置換されるグリフの数が多い場合、すなわち、変更されず「そのまま」の状態として残るグリフの数が少ない場合は、オプション「-g」を使った方が、ソースフォントから出力フォントへコピーするグリフの指定が簡単になります。さらに、併合フォントのマッピングファイルには、グリフ名または CID 番号が同じであっても、それを適切に指定しなければなりません。そうしないと、生成される出力フォントに、その名前または CID 番号によってグリフを含めることができません。

例えば、font.src 中の「c21」というグリフを、font.merge 中のグリフ「c21」と置き換えたい場合は、以下のコマンドを使います。

```
% mergeFonts -gx c21 font.out font.src map.merge font.merge
```

これによって、ソースフォント「font.src」の内容から「c21」という名前のグリフを除いた、新しい「font.out」というフォントが作られ、「c21」という名前のグリフが併合フォント「font.merge」から追加されます。この「map.merge」ファイルの内容は下記のようになります。

```
mergeFonts
c21      c21
```

1.2.5 新規グリフの追加

新しいグリフを追加するときは、オプション「-g」または「-gx」は不要です。また、グリフ名を変更せずに、併合フォントの全グリフをそのまま追加する場合は、マッピングファイルも不要です。併合フォント中のグリフの一部を追加する場合、または、グリフ名を変更する場合は、マッピングファイルが必要となります。以下の例を見てみましょう。

併合フォントにグリフ「c80」だけが含まれている場合（マッピングファイル不要）：

```
% mergeFonts font.out font.src font.merge
```

併合フォントにグリフ「c80」に加えてその他のグリフも含まれている場合（マッピングファイル必要）：

```
% mergeFonts font.out font.src map.merge font.merge
```

ファイル「map.merge」の内容は以下の通りとなります。

```
mergeFonts
c80      c80
```

1.2.6 CIDFont の作成

mergeFonts ツールは、複数の併合フォントを用いて有効な CIDFont ファイルを作成する目的でも利用できます。併合フォントは名前キー方式のフォントでも使用可能です。例えば、複数の名前キー方式のフォントを併合フォントに使い、グリフ名を適切な CID 番号に変換するマッピングファイルをオプション「-cid」と併用すると、何万というグリフを含んだ CIDFont を数秒で生成することができます。

以下のコマンドラインとマッピングファイルは、mergeFonts ツールを使うといかに簡単に CIDFont を作成できるかを示すものです。

```
% mergeFonts -cid cidfontinfo cidfont.ps NOTDEF.map NOTDEF.pfa \
hiragana.map hiragana.pfa katakana.map katakana.pfa r30.map r30.pfa \
r31.map r31.pfa
```

マッピングファイル「NOTDEF.map」の内容（単一グリフ : CID+0）:

```
mergeFonts KozMinPr6N-Heavy-Generic
0  NOTDEF
```

マッピングファイル「hiragana.map」の内容（グリフ数 83: CID 収容範囲 842 〜 924）:

```
mergeFonts KozMinPr6N-Heavy-Kana
842      smalla
843      a
…（この間の 79 行は省略）
923      wo
924      nn
```

マッピングファイル「katakana.map」の内容（グリフ数 87: CID 番号 660 および CID 収容範囲 925 〜 1010）:

```
mergeFonts KozMinPr6N-Heavy-Kana
660      extender
925      smalla
926      a
…（この間の 82 行は省略）
1009     smallka
1010     smallke
```

マッピングファイル「r30.map」の内容（グリフ数 94: CID 収容範囲 1125 〜 1218）:

```
mergeFonts KozMinPr6N-Heavy-Kanji
1125     c21
1126     c22
…（この間の 90 行は省略）
1217     c7D
1218     c7E
```

マッピングファイル「r31.map」の内容（グリフ数 94: CID 収容範囲 1219 〜 1312）:

```
mergeFonts KozMinPr6N-Heavy-Kanji
1219     c21
1220     c22
…（この間の 90 行は省略）
1311     c7D
1312     c7E
```

このように、指定のマッピングファイルと併合フォントを用いて上記のコマンドラインを実行すると、359 個のグリフを含む CIDFont（CID 番号：0、660、842 〜 1010、1125 〜 1312）が作成され、以下の 3 つのヒント辞書が作成されます。

```
KozMinPr6N-Heavy-Generic
KozMinPr6N-Heavy-Kana
KozMinPr6N-Heavy-Kanji
```

なお、マッピングファイル「r30.map」と「r31.map」の一行目には同じヒント辞書名が指定されています。ヒント辞書名を指定することで、作成される CIDFont のヒント辞書の構成を細かく調整することができます。また、複数の併合フォントの各マッピングファイルの一行目に同じヒント辞書名が指定されている場合、作成される CIDFont のヒント辞書は、同一のヒント辞書名を持つ最初の併合フォントからその値を継承します。

ヒント辞書名が指定されていない場合、併合フォントの各グリフは、併合フォントの「FontName」に従って名前の付けられた独自のヒント辞書に割当てられますが、これは正確さを欠き、能率的ではありません。

CIDFont のヒント辞書の調整を行うには細心の注意が必要ですが、mergeFonts を使って行うことができます。

1.3 rotateFont の使用法

このツールは、mergeFonts ほどにはではありませんが、名前キー方式と CID キー方式の両方のフォントに使えるうえ、グリフ名の変更、回転、シフト、さらにグリフの幅の変更に使用できます。グリフのシフトは X 軸と Y 軸の両軸について行います。グリフの回転と XY 軸についてのシフトは、コマンドライン上で指定して全体的にも行えますが、マッピングファイルを使って、XY 軸についてのシフトと、グリフ名及び字幅の変更を行うこともできます。

現行のアドビシステムズの文字コレクションでは、縦組用にあらかじめ回転させた形状の非全角幅のグリフを指定しており、「vrt2」GSUB フィーチャによってアクセス可能となっています。アドビシステムズの日本語用の文字コレクションで、あらかじめ回転させた形状に対応したのは、Adobe-Japan1-3 が最初のものでした。rotateFont ツールを使うと、オリジナルの（つまり、回転する前の）グリフ形状を回転させる工程が簡単になります。

1.3.1 コマンドライン

rotateFont ツールは、常に出力フォントの形式を指定するオプション、すなわち、Type 1 フォントにはオプション「-t1」、そして、CFF フォントにはオプション「-cff」を使って起動しなければなりません。いずれのの場合も、出力フォントが名前キー方式のフォントになるか、CID キー方式のフォントとなるかは、入力フォントの形式によって決まります（つまり、入力フォントと同じ形式のフォントが出力されます）。

rotateFont ツールは、mergeFonts ツールとは異なり、名前キー方式のフォントと CID キー方式のフォントの間の変換を行うことはできません。さらに mergeFonts ツールと異なる点は、コマンドライン上で指定する最初のフォントが入力フォントで、2 つ目のフォントが出力フォントになることです。また、変更をマッピングファイルで行うのか、それとも全体的に変更を行うのかによって、それぞれ「-rtf」または「-rt」オプションを使い分けます。

以下の例は、rotateFont を使って、時計回りに 90 度回転し、さらに X 軸と Y 軸についてそれぞれ 120 ユニットと 880 ユニットずつシフトさせるためのコマンドラインを示したもので、3 つの値はどれも、コマンドラインオプション「-rt」の引数となっています。

```
% rotateFont -t1 -rt 90 120 880 font.in font.out
```

1.3.2 マッピングファイル

フォント中のグリフ全部を回転したり、XY 軸に沿ってシフトする場合は、マッピングファイルを必要としません。マッピングファイルが必要となるのは、このファイルに記述されたグリフだけを出力フォントに含めたいときで、その際、マッピングファイル名はオプション「-rt」の後に記述します。ここで注意すべき点は、「.notdef」のグリフ定義が必要なことで、CID キーフォントの場合はマッピングファイル中に CID+0 が記されていなければなりません。

rotateFont のマッピングファイルには、以下に示すように、一行に 5 つの項目を含みます。

- 入力グリフ名または CID 番号

- 出力グリフ名または CID 番号
- グリフの幅：現在の幅をそのまま維持する場合は「None」を使用
- X 軸シフト：変更しない場合は「0（ゼロ）」、負の値はグリフを左方向へシフト
- Y 軸シフト：変更しない場合は「0（ゼロ）」、負の値はグリフを下方へシフト

グリフ名を変更しない場合は、入力グリフ名と出力グリフ名が同じになります。以下の例は、「exclam」というグリフをそのままの状態で、rotateFont を実行する際のマッピングファイルの内容を示したものです。

```
exclam exclam None 0 0
```

注： マッピングファイルが指定されている場合、そのファイルで指定されている XY 軸シフト値が、コマンド上のオプション「-rt」の引数値より優先されます。つまり、「-rt」と「-rtf」の両方のオプションを使用しなければならない場合、XY 軸のシフト値は、「-rt」の 2 つ目と 3 つ目の引数で指定するのはなく、マッピングファイル上で指定する必要があります。

注： 入力グリフ名と出力グリフ名のフィールドの順序は、mergeFonts のマッピングファイルの順序と逆になっています。つまり、mergeFonts と rotateFont のマッピングファイルで従うべき入力 / 出力グリフ名の順序は、それぞれのコマンドラインで指定された入力 / 出力フォントファイルの順序と同じになります。

1.3.3 グリフ名の変更

rotateFont ツールを使うと、マッピングファイルの 2 つ目のカラムに新規のグリフ名または CID 番号を指定することでグリフ名を簡単に変更することができます。以下の rotateFont マッピングファイルでは、グリフ名「exclam」が「c21」に変更されます。

```
exclam c21 None 0 0
```

グリフ名を変更しない場合は、最初の項目と 2 つ目の項目のグリフ名または CID 番号を同じにします。

1.3.4 グリフの回転

グリフの回転は、以下に示すように、スペースで区切られた３つの整数値を引数とするオプション「-rt」のみにより、rotateFont を実行できます。

- 時計回りに回転する度数：回転しない場合は「0（ゼロ）」を使用、負の値は時計と反対回り
- X 軸シフト：変更しない場合は「0（ゼロ）」
- Y 軸シフト：変更しない場合は「0（ゼロ）」

以下は、グリフを時計回りに 90 度回転し、X 軸について 200 ユニット、Y 軸について 800 ユニットだけシフトをする際のコマンドラインの例です。

```
% rotateFont -t1 -rt 90 200 800 font.in font.out
```

以下のコマンドラインは、出力フォントは上記とまったく同一のものとなりますが、負の値（-270 度）を指定して時計と反対回りに回転させた例です。

```
% rotateFont -t1 -rt -270 200 800 font.in font.out
```

なお、rotateFont では、グリフの原点である座標（0、0）を中心に回転を実行します。しかし、ラテンと非ラテン文字、CJK と非 CJK 文字のように、異なるグリフクラスではベースラインも違ってくるため、グリフを回転させた際に通常、XY 軸についてシフトさせて、全角の正方形の枠内で正しい位置に配置し直す必要が出てきます。

1.3.5 グリフの幅の変更

マッピングファイルの 3 カラム目の項目はグリフの幅を指定するためのものです。次のマッピングファイルでは、元の幅には関係なく、グリフの幅を 1000 ユニットに変更します。

```
exclam exclam 1000 0 0
```

グリフの幅を変更しない場合は、以下のように「None」と指定します。

```
exclam exclam None 0 0
```

1.3.6 グリフのシフト

グリフを全体的にシフトする場合は、オプション「-rt」を使ったコマンドラインで実行するのが最適です。オプション「-rt」の 2 つ目と 3 つ目の引数で、それぞれ X 軸および Y 軸のシフト値を指定します。特定のグリフだけをシフトしたい場合は、マッピングファイルを使い、その中の 4 カラム目と 5 カラム目の項目に、それぞれ X 軸および Y 軸のシフト値を指定します。次の例では、「exclam」という名のグリフを X 軸上で -100 ユニット（左方向）、Y 軸上で 250 ユニット（上方）にシフトします。

```
exclam exclam None –100 250
```

どちらか一方の軸を変更しない場合、両方とも変更しない場合は、以下のように、「0（ゼロ）」を指定します。

```
exclam exclam None 0 250
exclam exclam None –100 0
exclam exclam None 0 0
```

1.3.7 CID フォントの使用

rotateFont ツールでは、CID キー方式のフォントを入力フォントとして使用できます。グリフ名を指定する代わりに、CID 番号を使うわけです。以下のマッピングファイルでは、CID+1200 のグリフ幅を 1500 ユニットに変更し、さらに X 軸上で 250 ユニットだけシフトしています。こうすることで、元の幅が 1000 ユニットであった場合に、新しいグリフ幅の 1500 ユニットの中央にグリフを配置することができます。

```
0 0 None 0 0
1200 1200 1500 250 0
```

上記のマッピングファイルには必須項目である CID+0 が含まれている点に注目してしてください。

1.4 CIDFont の生成に関する注意点

rotateFont と mergeFonts のツールはともに、CIDFont に適用したり、CIDFont を作成する際に利用できますが、生成される CIDFont のヘッダはまだ不完全であるため、入念にテキスト処理を行う必要があります。例えば、FontInfo 辞書には、Notice、FullName、FamilyName、Weight、FSType といった項目を追加したり調整する必要のある場合があります。また、XUID 配列の設定も必要となります。さらに、CIDFontName、CIDFontVersion、CIDSystemInfo の辞書については、入力ファイルである「cidfontinfo」に基づいて、mergeFonts で、適切に設定しなければなりません。

現行のアドビシステムズの文字コレクションでは、あらかじめ回転させた形状の非全角幅のグリフを収録する CID 範囲を指定しています。これは、典型的にはプロポーショナルおよび半角のラテン文字グリフを含みますが、かならずしもそれらのグリフの種別に限定はされません。回転済みのグリフの CID 収録範囲についての詳細は、関連するアドビシステムズの文字コレクションのテクニカルノートをご参照ください。

http://www.adobe.com/devnet/font/

rotateFont は、回転前（直立）の形から回転したグリフを生成するのに理想的なツールです。回転したグリフを含めて生成されたフォントは、mergeFonts ツールへ入力する併合フォントとして使用可能で、多種多様なグリフを完備する CIDFont の構築に利用することができます。

1.5 autohint の使用法

autohint は、mergeFonts と rotateFont ツールで CIDFont を作成した後に利用できる、もう一つの非常に便利なツールで、AFDKO Version 2.0 またはそれ以降に搭載されています。このツールは、名前キー方式または CID キー方式のフォントに対して実行可能で、ヒンティングのパラメータ（BlueValues 配列、OtherBlues 配列、StemSnapH 配列、StemSnapV 配列、StdHW および StdVW）を使って、CharStrings に対してヒントを適用します。利用できるコマンドラインの種々のオプションを試してみて、最適なものを選択されることを推奨します。

1.5.1 並び域およびステム幅の値の算出

悪いヒンティングであれば、ヒンティングなどない方がよい、というのがレンダリングエンジンやヒンティングの専門家たちの考えです。同様に、不正確に算出したヒンティングパラメータは、autohint ツールを使用したとしても、悪いヒンティング結果を招きます。さらに、CharStrings のパスは、Type 1 CharString の仕様に準拠していなければなりません。例えば、コントロールポイント（与えられたヒンティングパラメータに基づき、ヒントによってどのステムを制御するかを決定するために用いられる）は極点の位置に配置しなければなりません。

並び域およびステム幅値の算出には細心の注意を払う必要があります。AFDKO Version 2.0 およびそれ以後のバージョンに搭載された stemHist ツールは、ヒストグラムのデータを生成して、並び域とステム幅値の算出を支援するものです。前者（並び域の算出）の機能は、コマンドライン上でオプション「-a」を使って実行します。

1.6 結果の検証

AFDKO Version 2.0 に含まれるツール「tx」は、グリフの概要を素早く生成するための便利なツールで、mergeFonts と rotateFont の各ツールを実行結果を検証します。その使用法は簡単で、オプション「-pdf」を使って、STDOUT 出力を PDF ファイルにリダイレクトするだけです。これらのツールを、数日から一週間ほど試してみて、その潜在的な能力を確認されることを推奨します。さらに多くの利用法があることを発見されることでしょう。これらのツールを使って実験を行って、tx ツールから出力される結果を検証することによって、満足のいく結果が得られるはずです。

AFDKO 2.0 版教程：mergeFonts、rotateFont 和 autohint

1.1 简介

AFDKO（Adobe Font Development Kit for Opentype）2.0 版于 2006 年下半年发布，它引入了三个功能极为强大的新字体开发工具，即 mergeFonts、rotateFont 和 autohint。这三个新工具配合细致灵巧的文字处理技术，可以用来建构功能全面的 CIDFont 汇总工具。当然，这些工具也可用于大量其他用途。

mergeFonts 工具可以处理按名称和 CID（Character ID）进行索引的字体，并且能执行下列任务：重命名字形、替换字形和添加字形。在命令行中允许同时输入多个字体文件，但是只能产生一个输出文件。rotateFont 工具也可以处理按名称和 CID 进行索引的字体，但它的主要功能是旋转和移动字形，也可用来重命名字形和更改字形宽度。autohint 工具可基于输入字体的 Hint 字典（对按 CID 进行索引的字体而言，有多个 Hint 字典）中的 Hint 参数（对齐区域和字干宽度的值）创建 Type 1 Hint，然后将其添加到 CharString。autohint 工具同样也可以处理按名称或 CID 进行索引的字体。

本教程面向使用 AFDKO 2.0 或更高版本的字体开发人员，与在命令行中输入“-u”和“-h”选项分别调出的用法介绍和帮助页面相比，本文对这三个工具的介绍更为深入。

最初的版本是用英文编写的，除了已有的中文和日文译本，该教程也将被翻译成韩文。每个版本的内容我们都力求简洁、清晰、易于理解。

如果对 mergeFonts、rotateFont 或 autohint 工具有任何疑问，或在使用工具时发现错误或其他奇怪的行为，请随时联系作者 Ken Lunde (lunde@adobe.com)。

注意：使用 mergeFonts 和 rotateFont 工具时，需要严格区分相应映射文件中的行尾标记。例如，在 Mac OS X 上使用“终端”(Terminal) 应用程序运行这两种工具时，映射文件必须使用 Unix 行尾标记 (LF; `0x0A`)，而不是 Macintosh 行尾标记 (CR; `0x0D`)。

注意：本文档介绍的 AFDKO 工具不能处理 Macintosh resource-fork font，而且 Type 1 字体必须是 Flat file（即 data-fork），可用 Fontographer 或 FontLab 等商业软件工具创建。FontLab Studio 5 将这种 Type 1 字体称为“ASCII/Unix Type 1”字体，使用“.pfa”作为默认的文件扩展名。

1.2 使用 mergeFonts

mergeFonts 工具的功能非常强大，它可以从一个或多个按名称或 CID 进行索引的源字体中提取多套字形，然后将所有提取的字形合并成一个按名称或 CID 进行索引的输出字体。

1.2.1 映射文件

当输入的单个或多个字体文件需要以某种方式划分子集或重命名字形时，mergeFonts 需要使用映射文件。文件的格式很简单，第一行必须按以下方式设置：

```
mergeFonts
```

该行明确地将文件标识为 mergeFonts 映射文件。当处理按 CID 进行索引的字体时，Hint 字典的名称可以直接被指定到关键字“mergeFonts”之后，如：

```
mergeFonts KozMinProVI-Heavy-Kanji
```

上面语句的作用是将指定的 Hint 字典分配到映射文件后续行中出现的 CID 中。如果 Hint 字典在 CIDFont 中不存在，将创建 Hint 字典。

在“mergeFonts”行下面的语句简单指定了一对字形名称或 CID，该对的第一个对象（第一列）是源（新）字形的名称或 CID，第二个对象（第二列）是添加 / 替换（原始或旧）字形的名称或 CID。如果不需要更改字形名称，则两个对象（列）应相同。下面是一个有效的 mergeFonts 映射文件的示例：

```
mergeFonts KozMinProVI-Heavy-Kanji
1125 c21
1126 c22
1127 c23
1128 c24
```

在上面的示例中，输入字体中的字形名称“c21”到“c24”分别被重命名为 CID 1125 到 1128，并且被分配到一个名为“KozMinProVI-Heavy-Kanji”的 Hint 字典。注意 CID 也可以使用五位数字即空位补零的整数表示方法，所以“1125”和“01125”都表示 CID=1125。

创建 mergeFonts 映射文件时，必须在至少一个映射文件中包含或指定“.notdef”（对于按名称进行索引的字体使用“.notdef”，对于按 CID 进行索引的字体则使用 CID=0）。

1.2.2 命令行

mergeFonts 工具通常可以运用命令选项处理多个输入文件。“-g”和“-gx”选项需要在后面列出一个或多个字形名称或 CID，它们仅适用于“源”字体，即命令行中多个输入字体文件中的第一个文件，后面的输入字体文件则都被看作“合并”字体。“-g”选项用于指定要从源字体复制到输出字体中的字形；“-gx”选项则用于指定要从源字体中排除的字形，也就是说这些被排除的字形将不被复制到输出字体。在替换字形时，这些选项非常有用，可以根据要替换字形的数量多少来选择合适的选项。例如，如果要替换的字形数量较少，使用“-gx”会更轻松，如果要替换的字形数量较多，也就是说不替换的字形的数量较少，使用“-g”会更简单。

通过 CID 指定字形时，“-g”和“-gx”选项可以使用 CID 范围作为参数。通过使用逗号和连字号来指定 CID 和 CID 范围。下面的参数指定了 CID 1、3 和 50 到 100：

```
1,3,50-100
```

当输出文件是 CIDFont 时，需要在“-cid”选项后面列出“cidfontinfo”文件的名称，所有映射文件都必须在第一列中指定 CID。使用“-h”选项调用 mergeFonts 可查看有效的“cidfontinfo”文件的格式以及示例。“cidfontinfo”文件的用途是提供 ROS（Registry、Ordering 和 Supplement）信息，以及其他 CIDFont 文件头的基本信息。

当需要使用映射文件时，在命令行中应将映射文件放在其对应的合并字体前面。例如，以下命令指定了一个输出文件、一个源字体和两个合并字体，每个合并字体都有独立的映射文件：

```
% mergeFonts font.out font.src map1.merge font1.merge map2.merge font2.merge
```

注意：默认情况下，在没有提供或没有指定映射文件时，将添加字体中的所有字形。

注意：如果名称发生冲突，将根据 mergeFonts 遇到的第一个字体在输出字体中添加字形，而不管它是源字体还是其中一个合并字体。

注意：输入字体文件可以是按名称进行索引和按 CID 进行索引的混和字体，但是它们都必须具有相同的 CharString 类型，明确来讲就是 Type 1 或 Type 2 (CFF)。

1.2.3 重命名字形

无论是重命名字体中的一个还是多个字形，都必须在映射文件中定义字体中的所有字形，只有那些要重命名的字形需要在第一列中指定不同于源字形的名称。下面的映射文件示例用来将字形名称从“exclam”更改为“c21”：

```
mergeFonts
c21 exclam
```

将字形名称转换为 CID 时，需要使用映射文件。下面的映射文件示例可以将字形名称“c21”、“c22”、“c23”和“c24”转换为 CID 1125 到 1128：

```
mergeFonts
1125 c21
1126 c22
01127 c23
01128 c24
```

请留意上例中用于指定 CID 的两种不同的记数法，其中一种是空位补零的五位数字。实际使用时，为保持前后一致应使用同一种记数法，以避免混淆并增强可读性。

1.2.4 替换字形

替换字形时，需要两个或多个输入字体。包含被替换字形的字体称为“源”字体，包含替换字形的字体称为“合并”字体。

在“-gx”选项后面列出要替换的字形名称或 CID，“-gx”仅适用于源字体，它指示 mergeFonts 复制除指定字形之外的所有字形。如果要替换的字形数量较多，也就是说要保留的字形数量较少，使用“-g”选项会更轻松，该选项用于指定要从源字体复制到输出字体中的字形。如果一些源字体中的字形通过”-gx”的设置没有被复制到输出字体中，那么在合并字体的映射文件中必须指定相同的字形名称或 CID，否则生成的输出字体将不包含具有该名称或 CID 的字形。

例如，如果要用 font.merge 中的字形“c21”替换 font.src 中的字形“c21”，请使用以下命令行：

```
% mergeFonts -gx c21 font.out font.src map.merge font.merge
```

这时将从源字体 (“font.src”) 中排除字形“c21”,然后添加合并字体 (“font.merge”) 中的字形“c21”，创建一个名为“font.out”的新字体。“map.merge”文件的内容如下：

```
mergeFonts
c21 c21
```

1.2.5 添加新字形

添加新字形时，不必使用“-g”或“-gx”选项。如果合并字体中的所有字形都按原样添加而且不更改字形名称，则不需要设置映射文件。如果要添加合并字体中字形的子集或者要更改字形名称，就需要使用映射文件。请留意以下示例。

仅包含字形“c80”的合并字体（不需要映射文件）：

```
% mergeFonts font.out font.src font.merge
```

除字形“c80”之外，还包含其他字形的合并字体（需要映射文件）：

```
% mergeFonts font.out font.src map.merge font.merge
```

“map.merge”文件的内容如下：

```
mergeFonts
c80 c80
```

1.2.6 创建 CIDFont

mergeFonts 工具可以使用多个合并字体创建有效的 CIDFont 文件。合并字体除了可以是 CID 字体也可以是按名称进行索引的字体。例如，可以将多个按名称进行索引的字体作为合并字体，通过使用“-cid”选项和将字形名称转换成相应 CID 的映射文件，快速生成包含成千上万种字形的 CIDFont。

通过下面的命令行和映射文件来说明如何轻松地使用 mergeFonts 工具创建 CIDFont：

```
% mergeFonts -cid cidfontinfo cidfont.ps NOTDEF.map NOTDEF.pfa \
hiragana.map hiragana.pfa katakana.map katakana.pfa r30.map r30.pfa \
r31.map r31.pfa
```

“NOTDEF.map”映射文件的内容（一个字形：CID=0）：

```
mergeFonts KozMinProVI-Heavy-Generic
0 NOTDEF
```

“hiragana.map”映射文件的内容（83 个字形：CID 842 到 924）：

```
mergeFonts KozMinProVI-Heavy-Kana
842 smalla
843 a
…（省略 79 行）
923 wo
924 nn
```

“katakana.map”映射文件的内容（87 个字形：CID 660 和 925 到 1010）：

```
mergeFonts KozMinProVI-Heavy-Kana
660 extender
925 smalla
926 a
…（省略 82 行）
1009 smallka
1010 smallke
```

“r30.map”映射文件的内容（94 个字形：CID 1125 到 1218）：

```
mergeFonts KozMinProVI-Heavy-Kanji
1125 c21
1126 c22
…（省略 90 行）
1217 c7D
1218 c7E
```

“r31.map”映射文件的内容（94 个字形：CID 1219 到 1312）：

```
mergeFonts KozMinProVI-Heavy-Kanji
1219 c21
1220 c22
…（省略 90 行）
1311 c7D
1312 c7E
```

使用指定的映射文件和合并字体运行以上命令行时，会创建包含 359 个字形的 CIDFont。涉及的 CID 包括 0、660、842 到 1010 和 1125 到 1312，而且还会创建总共三个 Hint 字典，如下所示：

```
KozMinProVI-Heavy-Generic
KozMinProVI-Heavy-Kana
KozMinProVI-Heavy-Kanji
```

请留意“r30.map”和“r31.map”映射文件第一行中指定的 Hint 字典名称是相同的。通过指定 Hint 字典名称可以严格控制最终 CIDFont 文件的 Hint 字典结构。另外，如果多个合并字体的映射文件在第一行中指定 Hint 字典名称相同，最终 CIDFont 的 Hint 字典将从具有该 Hint 字典名称的第一个合并字体中继承。

如果未指定 Hint 字典名称，每个合并字体的字形将分配到一个唯一的、按照各自合并字体的“FontName”命名的 Hint 字典，这种做法不准确，效率也不高。

必须严格控制 CIDFont 文件的 Hint 字典的结构，mergeFonts 工具是进行此类控制的工具。

1.3 使用 rotateFont

尽管 rotateFont 功能不如 mergeFonts 工具强大，但是 rotateFont 工具同样可以处理按名称和 CID 进行索引的字体，并可以执行重命名、旋转、移动字形及更改字形宽度的操作。字形移动是沿 X 轴和 Y 轴进行的，如果是对全部字形进行操作，可以通过在命令行中直接设置旋转和沿 X 轴和 Y 轴移动的参数来实现，也可以在映射文件中设定沿 X 轴和 Y 轴移动的参数、更改的字形名称及宽度值。

当前 Adobe character collections 为部分非全角字形指定了用于竖排的预旋转字形，可通过 GSUB 的“vrt2”特性来实现。对于日文字体，“Adobe-Japan1-3”是第一个包含这些预旋转字形的 Adobe character collections。rotateFont 工具简化了从未旋转的原始形态创建预旋转字形的流程。

1.3.1 命令行

rotateFont 工具通过选项的设置来指定输出字体的格式，Type1 字体使用“-t1”，CFF 字体使用“-cff”。在任何一种情况下，输出字体都将与输入字体的属性保持一致，即是按名称还是按 CID 进行索引。

与 mergeFonts 工具不同，rotateFont 工具不能在按名称和按 CID 进行索引的字体之间进行转换。另外一个不同点是命令行中指定的第一个字体是输入字体，而第二个字体是输出字体。是按映射文件执行操作还是对整个字体进行操作，这需要通过设置相应的“-rtf”或“-rt”选项来实现。

下面的 rotateFont 命令行示例会执行 90 度顺时针旋转，并沿 X 轴和 Y 轴分别移动 120 和 880 个单位，这三个数值都是“-rt”命令行选项的参数：

```
% rotateFont -t1 -rt 90 120 880 font.in font.out
```

1.3.2 映射文件

当字体中的所有字形都执行字形旋转和沿 X 轴和 Y 轴移动操作时，不需要映射文件。当个别字形需要执行这些操作时，就需要将这些字形定义在映射文件中，映射文件的名称放在“-rtf”选项的后面。注意“.notdef”字形条目是必需有的，如果是按 CID 进行索引的字体，则映射文件中必须有 CID=0 条目。

rotateFont 映射文件每行包含五列，如下所示：

- 输入字形的名称或 CID
- 输出字形的名称或 CID
- 字形宽度—使用“None”按原样传递宽度
- X 轴偏移量—设置为“0”（零）时不变化；使用负数时，字形向左移动
- Y 轴偏移量—设置为“0”（零）时不变化；使用负数时，字形向下移动

如果字形名称保持不变，那么输入字形和输出字形的名称应相同。下面是 rotateFont 映射文件语句的示例，将按原样传递名为“exclam”的字形：

```
exclam exclam None 0 0
```

注意：映射文件中的 X 和 Y 轴偏移量要比为“-rt”命令行选项指定的参数优先级高。换句话说，如果必须使用“-rt”和“-rtf”这两个选项，则应在映射文件中指定 X 和 Y 轴偏移量，而不是将它们作为“-rt”选项的第二和第三个参数。

注意：与 mergeFonts 映射文件相比，输入字形和输出字形名称字段的顺序相反。换句话说，mergeFonts 和 rotateFont 映射文件中的输入 / 输出字形名称的顺序和各自命令行中指定的输入 / 输出字体文件顺序一致。

1.3.3 重命名字形

通过在映射文件的第二列指定新字形名称或 CID，可以使用 rotateFont 工具轻松重命名字形。下面的 rotateFont 映射文件语句将字形名称从“exclam”更改为“c21”：

```
exclam c21 None 0 0
```

如果不需要对字形进行任何更改，第一列和第二列中的字形名称或 CID 应保持一致。

1.3.4 旋转字形

字形旋转只能通过对 rotateFont 使用“-rt”选项来执行，它需要三个以空格分隔的整数作为参数，如下所示：

- 顺时针旋转的度数—“0”（零）表示不旋转；负数表示逆时针旋转
- X 轴偏移量—“0”（零）表示不沿 X 轴移动
- Y 轴偏移量—“0”（零）表示不沿 Y 轴移动

下面的命令行示例将字形顺时针旋转 90 度，然后将字形沿 X 轴方向移动 200 个单位，沿 Y 轴方向移动 800 个单位：

```
% rotateFont -t1 -rt 90 200 800 font.in font.out
```

下面的命令行会产生相同的输出字体，不同之处是通过使用负数值进行逆时针旋转而获得的：

```
% rotateFont -t1 -rt -270 200 800 font.in font.out
```

请注意，rotateFont 以坐标（0,0）为字形原点进行旋转。由于不同字形类的基线不同（如 Latin 和非 Latin，或 CJK 和非 CJK），通常还需要进行 X 轴和 Y 轴的偏移，以便重新将旋转后的字形放在字形框中。

1.3.5 更改字形宽度

映射文件的第三列用于指定字形宽度。下面的映射文件语句会将字形宽度更改为 1000 个单位，而不管原始宽度是多少：

```
exclam exclam 1000 0 0
```

如果宽度需要保持原样不变，应使用“None”，如下例所示：

```
exclam exclam None 0 0
```

1.3.6 字形移动

最好在命令行中使用“-rt”选项来执行全局范围的字形移动。“-rt”选项的第二和第三个参数用于指定 X 轴和 Y 轴偏移量。如果只需移动特定的字形，应使用映射文件，并在第四和第五列中分别指定 X 轴和 Y 轴偏移量。下面的示例将字形“exclam”沿 X 轴方向移动 –100 个单位（向左），沿 Y 轴方向移动 250 个单位（向上）：

```
exclam exclam None –100 250
```

如果不需要移动，请为其中一个轴或两个轴同时使用“0”（零），如下所示：

```
exclam exclam None 0 250
exclam exclam None –100 0
exclam exclam None 0 0
```

1.3.7 处理 CIDFont

rotateFont 工具的输入字体可以使用按 CID 进行索引的字体，这时不是指定字形名称，而是指定 CID。假设字形的原始宽度是 1000 个单位，下面的 rotateFont 映射文件会将 CID=1200 的字形的宽度设为 1500 个单位，同时沿 X 轴将字形移动 250 个单位，从而使字形在新的 1500 单位宽度内居中：

```
0 0 None 0 0
1200 1200 1500 250 0
```

请注意在映射文件中包含 CID=0 条目。

1.4 创建 CIDFont 的注意事项

尽管 rotateFont 和 mergeFonts 工具可以处理 CIDFont 并且创建 CIDFont，但是生成的 CIDFont 头信息是不完整的，需要进行仔细的文字处理。例如，以下的 FontInfo 字典条目可能需要进行添加或调整：Notice、FullName、FamilyName、Weight 和 FSType，还需要设置“XUID”数组。mergeFonts 工具会根据提供的“cidfontinfo”文件内容设置相应的 CIDFontName、CIDFontVersion 和 CIDSystemInfo 字典。

全角字形的 CID 不包含在当前 Adobe character collection 的预旋转字形 CID 范围之内，一般只包括变宽和半宽的 Latin 字形，但这并不意味着只能局限于这类字形。有关预旋转字形的 CID 范围的详细信息，请参阅每种 Adobe 字符集相应的 Adobe 技术说明：http://partners.adobe.com/public/developer/font/

rotateFont 工具最适合为从未旋转（直立）的字形创建预旋转字形并保存到输出字体中。该生成字体可以用作 mergeFonts 工具所需的合并字体（输入字体），以创建完整的 CIDFont。

1.5 使用 autohint

使用 mergeFonts 和 rotateFont 工具创建完 CIDFont 之后，可以使用另一个非常有用的工具 autohint(仅在 AFDKO 2.0 或更高版本中才提供)。autohint 工具可以处理按名称或 CID 进行索引的字体，并基于 Hint 参数 (BlueValues 数组、OtherBlues 数组、StemSnapH 数组、StemSnapV 数组、StdHW 和 StdVW) 将 Hint 控制应用到 CharString。您可以研究其命令行选项来确定最适合的选项。

1.5.1 计算对齐区域及字干宽度的值

按照 rendering-engine 和 Hint 方面专家的说法，没有 Hint 控制的效果会好于差的 Hint 控制。同样，当使用 autohint 工具时，不恰当的 Hint 参数设置会导致较差的 Hint 效果。另外，CharString 的 Path 操作应该符合 Type 1 CharString 规格。例如，以 Hint 为目的的控制点，由于它需要特别去判断和确定什么样的字干需要用 Hint 来控制，所以 autohint 会基于提供的 Hint 参数，将该控制点设置在极值点。

强烈建议仔细计算对齐区域及字干宽度的值。AFDKO 2.0 版或更高版本中包含的 stemHist 工具可以产生直方图数据，利用这些统计数据来协助计算对齐区域及字干宽度的值。使用“-a”命令行选项调用该功能。

1.6 验证结果

AFDKO 2.0 版中包含的 tx 工具可以快速生成字形预览文件，对于验证 mergeFonts 和 rotateFont 工具的生成结果非常有用。生成字形预览文件的方法很简单，只需要使用“-pdf”选项，并将 STDOUT (Standard Output stream) 直接输出到 PDF 文件即可。建议您最好花几天或整个一周时间来研究这些工具，充分熟悉其功能，相信您将发现这些字体开发工具的其他用途。不断尝试这些工具，然后使用 tx 工具检查输出结果，从而得到令人满意的结果。

2007 年 2 月 21 日