



# Workshop: Manipulating CID-Keyed Fonts Using AFDKO Tools

Dr. Ken Lunde | Senior Computer Scientist | Adobe Systems Incorporated



# About The Sample Data For This Workshop...

- The sample data is based on Adobe's *Kozuka Gothic® Pr6N M* font
  - Included are 234 glyphs for Latin, punctuation, kana, and ideographs (kanji)
- Scripts for preparing sample data based on your own font are provided
- Copy an Adobe-Japan1-x "cidfont.ps" file into the PREP/AJ10 directory  
tx -t1 <font>.otf cidfont.ps
  - Execute the "build.sh" script in the PREP/AJ10 directory
    - Edit the /FontName parameters of {dingbats,generic,hiragana,kanji,katakana,proportional}.pfa
    - Copy the above \*.pfa files into CID-AJ10/1\_2\_CID-N-NOMAP and CID-AJ10/1\_3\_CID-N-MAP
    - Copy "font.pfa" into CID-AJ10/1\_1\_CID-1-NAME2CID, CID-AJ10/1\_4\_CID-1-CID2CID & PREP/AI0
    - Execute the "build1.sh" script in the CID-AJ10/1\_4\_CID-1-CID2CID directory
  - Execute the "build.sh" script in the PREP/AI0 directory
    - Copy {dingbats,generic,hiragana,kanji,katakana,proportional}-.\*.pfa into CID-AI0/1\_2\_CID-N
    - Copy "font-uni.pfa" into CID-AI0/1\_1\_CID-1

# Why Develop CID-Keyed Fonts?

- CID-keyed fonts support multiple FArray elements
  - Every CID is assigned to an FArray element
  - Each FArray element can have its own hinting parameters
    - /BlueValues, /OtherBlues, /StdHW, /StdVW, /StemSnapH, /StemSnapV, and so on
  - Up to 256 FArray elements can be included
  - In general, a separate FArray element is used for each script
- Mapping from an encoding to CIDs is controlled by CMap resources
  - Unicode uses UTF-32 CMap resources
- Each glyph is associated with a simple integer value
  - CID (Character ID)

# What Is AFDKO?

- AFDKO is an abbreviation for *Adobe® Font Development Kit for OpenType®*
- Almost all AFDKO tools support CID-keyed fonts

*tx*

*mergeFonts*

*rotateFont*

*stemHist*

*autohint*

*makeotf*

and so on...

# Three Very Important Workshop Takeaways

- Almost all AFDKO tools support CID-keyed fonts
- CID-keyed fonts should have more than a single FDArray element
  - And that it is relatively easy to control FDArray elements
- All of the techniques that are demonstrated during this workshop will scale...

# Three Very Important Workshop Takeaways

- Almost all AFDKO tools support CID-keyed fonts
- CID-keyed fonts should have more than a single FDArray element
  - And that it is relatively easy to control FDArray elements
- All of the techniques that are demonstrated during this workshop will scale...
  - ...to handle thousands or tens of thousands of glyphs

# Useful Command Lines For AFDKO-Based Font Development

- Glyph synopses are easily generated using the AFDKO *tx* tool

```
tx -pdf <font> glyphs.pdf
```

```
tx -pdf -g <glyphs> <font> glyphs.pdf
```

- Displaying the CIDs of a CID-keyed font (a filter for the *tx* tool)

```
extract-cids.pl <font>
```

```
extract-cids.pl -r <font>
```

```
extract-cids.pl -r -s <font>
```

- Displaying the glyph names of a name-keyed font (also a filter for the *tx* tool)

```
extract-names.pl <font>
```


# Useful Command Lines For AFDKO-Based Font Development (cont'd)

- Displaying the GIDs of a CID- or name-keyed font (also a filter for the *tx* tool)  
`extract-gids.pl <font>`  
`extract-gids.pl -r <font>`
- Displaying the FDArray element assignment (also a filter for the *tx* tool)  
`fdarray-check.pl <font>`
- About the command lines that will be demonstrated during this workshop...
  - For your convenience, and for learning, there are "build\*.sh" scripts in each directory
  - Of course, be sure to confirm the contents of each script prior to execution



# CID Versus GID

- CIDs and GIDs are identical for fonts that include all CIDs of an ROS
  - GIDs are always contiguous
  - When CIDs are not contiguous, such fonts are referred to as "subset" fonts
  - "ROS" is an abbreviation for /CIDSystemInfo's /Registry, /Ordering & /Supplement
- A slash (/) prefix is recommended for explicitly specifying glyphs by their CID  
CID+1200 → /1200
- The output of *extract-cids.pl* and *fdarray-check.pl* use the slash prefix
- Glyph synopses generated by *tx* show GIDs and CIDs
  - The GID is in the upper-left corner
  - The CID is in the lower-left corner, and uses a backslash (\) prefix

0,0	1,1	2,1	3,1	4,1	5,1	6,1	7,1	8,1	9,1	10,1
1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
	一	九	五	三	四	七	十	二	八	六
\0	\1200	\1757	\1938	\2174	\2203	\2275	\2375	\3275	\3392	\4065

# The All-Important “cidfontinfo” File

- The *mergeFonts* and *makeotf* tools use this file
  - The *makeotf* tool doesn't necessarily require this file
    - The same information can be specified through the use of *makeotf* command-line options
  - The *mergeFonts* tool requires this file when generating a CID-keyed font
- The “cidfontinfo” file lines below are specific to the *mergeFonts* tool:

<b>FontName</b>	(KozGoAJ10-Medium)
<b>FullName</b>	(Kozuka Gothic AJ10 OpenType Medium)
<b>FamilyName</b>	(Kozuka Gothic AJ10 OpenType)
<b>Weight</b>	(Medium)
<b>version</b>	(1.000)
<b>Registry</b>	(Adobe)
<b>Ordering</b>	(Japan1)
<b>Supplement</b>	0
<b>XUID</b>	[1 11 9273884]
<b>AdobeCopyright</b>	(Copyright 2001-2012 Adobe Systems Incorporated. All...)
<b>Trademark</b>	(Kozuka Gothic is either a registered trademark or...)

# The All-Important “cidfontinfo” File (cont'd)

- The “cidfontinfo” file lines below are specific to the *makeotf* tool:

<b>IsBoldStyle</b>	false # The same as the “-nb” option
<b>IsItalicStyle</b>	false # The same as the “-ni” option
<b>PreferOS/2TypoMetrics</b>	true # The same as the “-osbOn 7” option
<b>IsOS/2WidthWeighthSlopeOnly</b>	true # The same as the “-osbOn 8” option
<b>IsOS/2OBLIQUE</b>	false # The same as the “-osbOff 9” option
<b>UseOldNameID4</b>	false # The same as the “-newNameID4” option
<b>LicenseCode</b>	ADOBE # The same as the “-lic ADOBE” option

# About The “cidfontinfo” File’s /XUID Array...

- Correctly setting the “XUID” line of the “cidfontinfo” file
  - An /XUID array can contain up to four elements
  - The minimum number of elements is two
- The first element is set as the developer’s “Font XUID” value (an integer)
  - Adobe’s Font XUID value is “1”
- The subsequent elements are set by the developer
  - Each font should have a unique /XUID array
  - Different versions of the same font can use the same /XUID array
- The URL for Font XUID registration is below:

*[http://partners.adobe.com/public/developer/font/register/xuid\\_reg.do](http://partners.adobe.com/public/developer/font/register/xuid_reg.do)*

# AFDKO *mergeFonts* Tool Basics

- The *mergeFonts* tool combines multiple fonts into a single font resource
- A "cidfontinfo" file is necessary for name-keyed → CID-keyed conversion
  - The "-cid cidfontinfo" option and its argument must be specified
- When multiple source fonts include the same glyph, the first one is used
  - The "-gx <glyphs>" option is used to explicitly exclude glyphs in the first source font
- CIDs and glyph names can be changed by using *mergeFonts* mapping files
  - The first line of a *mergeFonts* mapping file must begin with "mergeFonts"
  - Glyph names of the "cid" + CID pattern are converted to CIDs without a mapping file
- FArray element names can be specified for CID-keyed fonts
  - This is specified on the first line as the first argument of "mergeFonts"  
mergeFonts KozGoAJ10-Medium-Kanji 1
  - The first argument is the FArray name; the second is the /LanguageGroup (0 or 1)

# The AFDKO *mergeFonts* Tool & FDArray Element Assignment

- CID-keyed fonts can include one or more FDArray elements
  - The maximum number of FDArray elements is 256; the minimum is one
- Every CID must be assigned to an FDArray element
- Each FDArray element has a /FontName parameter
  - The /FontName parameter of FDArray elements use the /CIDFontName as their base
    - This is specified by /CIDFontName + "-" + an identifier
- There are two methods for controlling FDArray element assignment
  - Inherit the name-keyed source font's /FontName parameter  
/FontName /KozGoAJ10-Medium-Kanji def
  - Specify as the first argument of the first line of the *mergeFonts* mapping file  
mergeFonts KozGoAJ10-Medium-Kanji 1
  - **Important Note: The latter method overrides the former method**

# Which ROS Is Best? Public ROSes or Adobe-Identity-0?

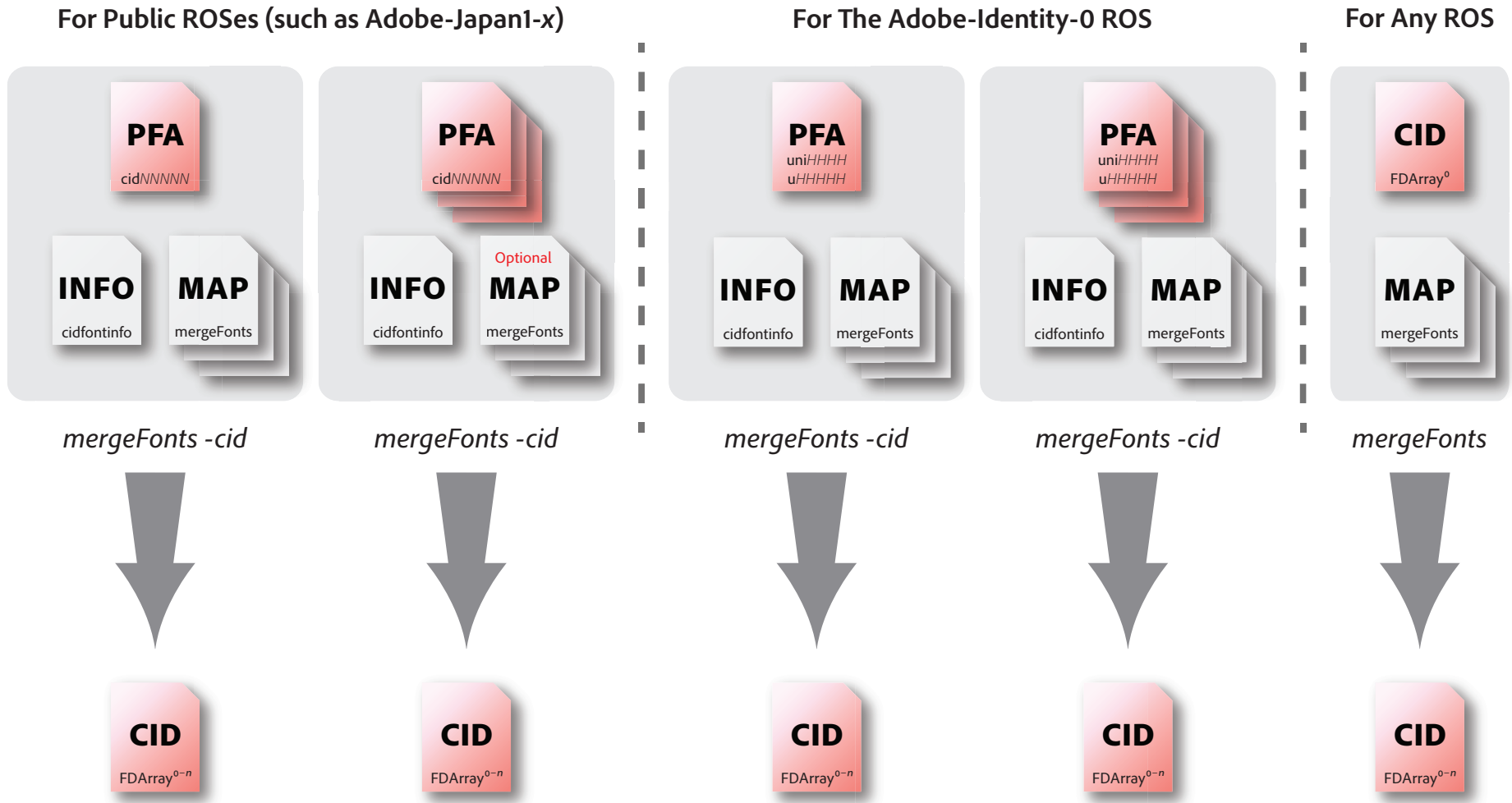
- Adobe-Identity-0 must be used when the glyphs are not in a public ROS
  - Example: The new “JIS” mark
    - This glyph is not included in Adobe-Japan1-6, but is an Adobe-Japan1-7 candidate
  - Example: Many of the vertical glyphs in Adobe's *Kazuraki*® font
    - Ideographs, standard-size kana, and hiragana ligatures
- When the glyphs are in a public ROS, either ROS can be used

# Which ROS Is Best? Public ROSes or Adobe-Identity-0? (cont'd)

- Public ROS Pros
  - Existing font-development materials can be leveraged
    - CMap resources and GSUB feature definitions
- Adobe-Identity-0 Cons
  - A font-specific UTF-32 CMap resource must be made
  - Font-specific GSUB features must be made



# AFDKO-Based CID-Keyed Font Development Workflows: *mergeFonts*



# Name-Keyed Fonts → Public ROS CID-Keyed Font

- There are two ways in which the name-keyed font data can be arranged
  - All of the glyphs are in a single name-keyed font resource
    - Used when the number of glyphs is small
  - The glyphs are in multiple name-keyed font resources
    - Used when the number of glyphs is large
- The glyph names adhere to appropriate CIDs of a Public ROS: "cid" + CID  
CID+1200 → cid1200
- For multiple font resources, *mergeFonts* mapping files are not necessary
  - The font resources' /FontName parameters must match the FArray element names

# Name-Keyed Fonts → Adobe-Identity-0 CID-Keyed Font

- There are two ways in which the name-keyed font data can be arranged
  - All of the glyphs are in a single name-keyed font resource
    - Used when the number of glyphs is small
  - The glyphs are in multiple name-keyed font resources
    - Used when the number of glyphs is large
- The glyph names correspond to Unicode scalar values: "uni" + Unicode value  
U+4E00 → uni4E00
- Non-BMP Unicode values use a shorter "u" prefix  
U+20000 → u20000
- *mergeFonts* mapping files must be prepared

# Splitting A Single FArray Element Into Multiple FArray Elements

- *mergeFonts* mapping files are necessary
  - At least one *mergeFonts* mapping file is required for each FArray element
  - Each FArray element must have a unique /FontName parameter
    - This is specified as the first argument on the first line of a *mergeFonts* mapping file
- The "-cid" option and the "cidfontinfo" file are not necessary
- The following is the command line:  
`mergeFonts <newfont> <farray-0-map> <font> ... <farray-n-map> <font>`

# AFDKO *rotateFont* Tool Basics

- In addition to rotating glyphs, the *rotateFont* tool can also perform the following:
  - Change glyph widths
  - Change glyph names or CIDs
  - Independently adjust X- and Y-axis glyph positioning
- Four operations are used to create the glyphs for the 'vrt2' GSUB feature
  - 90° clockwise rotation, make full-width, change CIDs, adjust glyph positioning
- 90° clockwise rotation is specified as the first argument of the "-rt" option  
-rt 90 0 0
- The other operations can be described in a file that is specified by the "-rtf" option
  - Input glyph, output glyph, glyph width, X-axis shift, Y-axis shift
  - Example: 1 8720 1000 120 880
- Command line example:  
`rotateFont -t1 -rt 90 0 0 -rtf <mapping_file> <input_font> <output_font>`

# Directly Hinting CID-Keyed Fonts (/StdHW & /StdVW)

- The *fdarray-check.pl* tool lists the CIDs assigned to each FDArray element  
`fdarray-check.pl <font>`
- The CIDs of an FDArray element are specified after the *stemHist* "-g" option  
`stemHist -all -g /633-/635,/638,/686-/687,/7887-/7888,/7911-/7912 <font>`
  - The output of the *stemHist* tool are the following two files:  
`<font>.hstm.txt` (horizontal stem widths)  
`<font>.vstm.txt` (vertical stem widths)
- Search for the highest-frequency stem values in these histogram files
- Specify these values in the "hintparam.txt" file for all FDArray elements

## Dingbats

```
/BlueValues [-250 -250 1100 1100] def  
/StdHW [69] def  
/StdVW [69] def
```

# Directly Hinting CID-Keyed Fonts (/BlueValues)

- The CIDs of an FArray element are specified after the *stemHist* "-a -g" options  
*stemHist -a -g /1,/6,/13,/15,/17-/26,/34-/59,/66-/91 <font>*
- The output of the *stemHist* tool are the following two files:
  - <font>.bot.txt* (bottom stem zones)
  - <font>.top.txt* (top stem zones)
- The first value-pair is the baseline and its overshoot from *<font>.bot.txt*
- The other value-pairs are the x-height, cap-height, and overshoots from *<font>.top.txt*
- A fixed /BlueValues array is recommended for non-Latin FArray elements  
*/BlueValues [-250 -250 1100 1100] def*
- Specify these values in the "hintparam.txt" file for all FArray elements

## Proportional

```
/BlueValues [-11 0 551 563 765 777] def  
/StdHW [93] def  
/StdVW [116] def
```

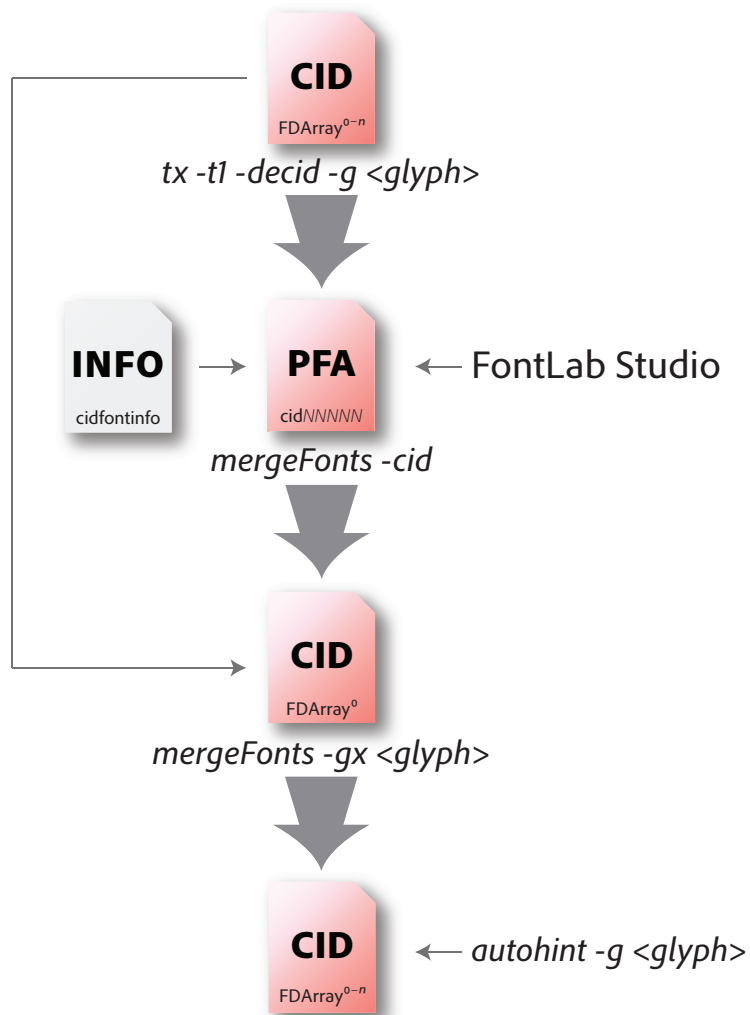
## Directly Hinting CID-Keyed Fonts (cont'd)

- Change the hinting parameters of each FDArray element using *hintcidfont.pl*  
`hintcidfont.pl hintparam.txt < cidfont-nohint.ps > cidfont-hint.ps`
- Use *tx* to confirm the hinting parameters of each FDArray element  
`tx -0 cidfont-hint.ps`
- Finally, execute the *autohint* tool to apply the hinting parameters  
`autohint -r -q -o cidfont.ps cidfont-hint.ps`

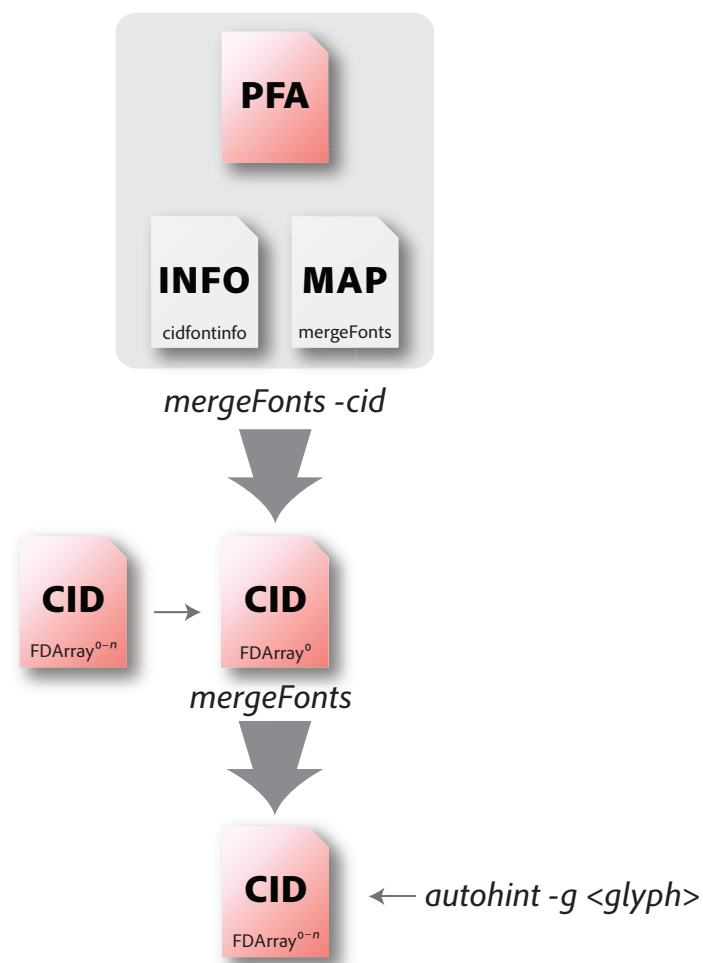


# AFDKO-Based CID-Keyed Font Development Workflows: Change/Add

## Changing Glyphs



## Adding Glyphs



# Changing Glyphs In CID-Keyed Fonts

- Use "tx -t1 -decid -g <glyphs>" to extract the glyphs to be changed
  - You can specify an FDArray element using the "-usefd <index>" option  
tx -t1 -decid -usefd 3 -g /2520 cidfont.ps cid2520.pfa
  - The result is a name-keyed font that can be easily edited
  - The glyph can now be changed by using a font editor, such as FontLab Studio
    - When generating the font, be sure to use "ASCII/UNIX Type 1" as the format
- Use *mergeFonts* to convert the modified glyphs into a CID-keyed font  
mergeFonts -cid cidfontinfo cid2520.ps cid2520.pfa
- Use *mergeFonts* to replace the original glyphs with the modified ones  
mergeFonts -gx /2520 cidfont-mod.ps cidfont.ps cid2520.ps
- Use *autohint* to apply the hinting parameters to the modified glyphs  
autohint -g /2520 -r -q -o cidfont.ps cidfont-mod.ps

# Adding Glyphs To CID-Keyed Fonts

- Use *mergeFonts* to convert the additional glyphs into a CID-keyed font
  - Be careful about FDArray element assignment!

```
mergeFonts KozGoAl0-Medium-Dingbats 1
```

```
0 .notdef
```

```
300 NewJIS
```

```
mergeFonts -cid cidfontinfo cid300.cid newjis.map newjis.pfa
```

- Use *mergeFonts* to combine the original font and the glyphs to be added

```
mergeFonts cidfont-add.ps cidfont.ps cid300.cid
```

- Use *autohint* to apply the hinting parameters to the additional glyphs

```
autohint -g /300 -r -q -o cidfont.ps cidfont-add.ps
```

# Correcting The FontBBox Array

- CID-keyed fonts made with *mergeFonts* may have inaccurate FontBBox arrays
  - The same is true of CID-keyed fonts that are made using *rotateFont*
- The *fix-fontbbox.pl* tool can be used to correct FontBBox arrays

```
fix-fontbbox.pl cidfont.ps > cidfont-fix.ps  
mv cidfont-fix.ps cidfont.ps
```

# Operations To Perform When Modifying CID-Keyed Fonts

- When modifying any font, it is *good practice* to increment the version
  - The *version-up-cidfont.pl* tool increments the version in a CID-keyed font in 2 places

```
%%Version: 1
/CIDFontVersion 1 def
```
  - The following is the command line:

```
version-up-cidfont.pl < cidfont.ps > cidfont-new.ps
```
  - The following is the result after executing *version-up-cidfont.pl*:

```
%%Version: 1.001
/CIDFontVersion 1.001 def
```
- Don't forget to increment the "version" line in the "cidfontinfo" file!

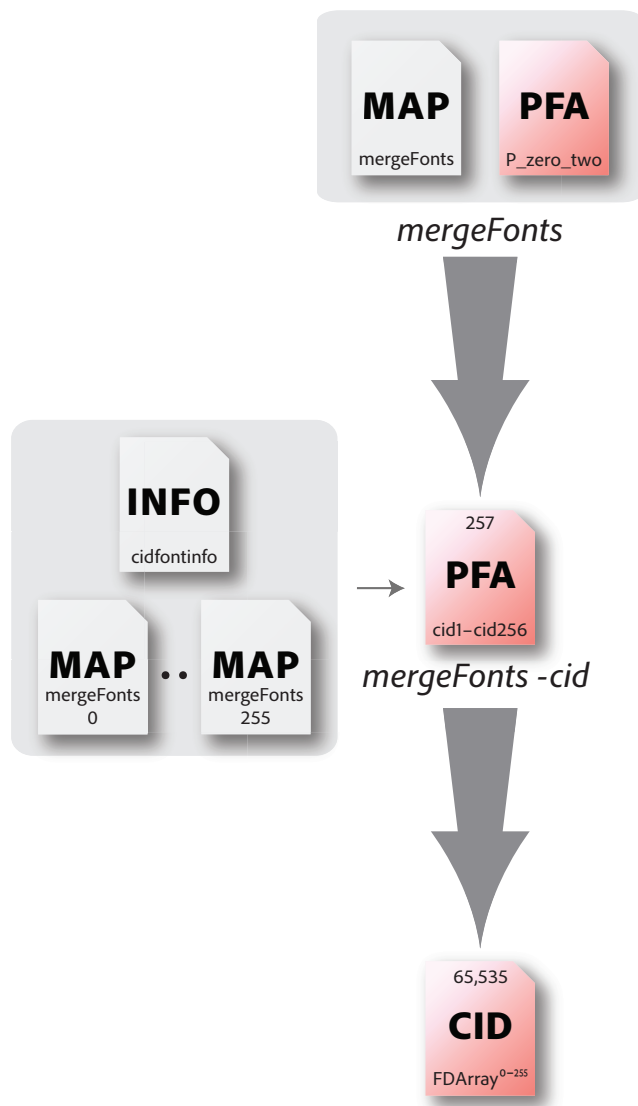
# Developing Adobe-Identity-0 UTF-32 CMap Resources

- Most of the data resides in the *mergeFonts* mapping files  
184 uni56FD
- This becomes the following UTF-32 CMap resource mapping  
<000056FD> 184
- Edit the "cmap-template.txt" file
  - Insert all of the mappings immediately after the "0 begincidchar" line  
0 begincidchar  
<00000020> 1  
...  
<000056FD> 184  
...  
endcidchar
- Execute the *cmap-tool.pl* tool
  - `cmap-tool.pl < cmap-template.txt > <cmap>`

# Converting CID-Keyed Fonts To CFF

- The end-game for a CID-keyed font is to build an OpenType/CFF font
  - A well-structured CID-keyed font results in a better OpenType/CFF font
- There are two tools that can convert a CID-keyed font to CFF: *tx* and *makeotf*
- The *makeotf* tool is recommended
  - A CID-keyed font serves as one of its input files via the "-f" command-line option
  - Subroutinization is possible via the "-r" or "-S" command-line options
    - The use of the "-r" command-line option implies the "-S" command-line option
- The *tx* tool can perform the same CID-keyed font → CFF function
  - The "-cff" command-line option is used
  - The *sfntedit* tool can be used to inject a CFF into an existing OpenType/CFF font  
`sfntedit -a CFF=<cff> <opentype_font>`
  - Subroutinization is possible via the "+S" command-line option

# AFDKO-Based CID-Keyed Font Development Workflows: Maximum Limits





# A CID-Keyed Font With The Maximum Glyphs & FArray Elements

- The maximum number of glyphs in a CID-keyed font is 65,535 (64K)
  - CIDs 0 through 65534
  - At the other end of the spectrum, the minimum number of glyphs is one
    - CID+0 (the so-called ".notef" glyph)
- A 257-glyph name-keyed font is created with one *mergeFonts* mapping file  
The glyphs are named ".notdef" and "cid1" through "cid256"
- A 64K-glyph CID-keyed font is created with 256 *mergeFonts* mapping files  
mk64k256fdarray.pl UnicodeP02 > build2.sh
- The command line is very long, so a "build2.sh" script is created and executed  
sh ./build2.sh
- The CID-keyed font includes 65,535 glyphs and 256 FArray elements

# Useful URLs

- AFDKO

*<http://www.adobe.com/devnet/opentype/afdko.html>*

- Adobe's CJK Type Blog

*<http://blogs.adobe.com/CCJKType/>*

- Font-related Adobe Technical Notes

*<http://www.adobe.com/devnet/font.html>*

- Font XUID Registration

*[http://partners.adobe.com/public/developer/font/register/xuid\\_reg.do](http://partners.adobe.com/public/developer/font/register/xuid_reg.do)*

- "CMap Resources" open source project

*<http://sourceforge.net/adobe/cmap/>*

- OpenType Specification

*<http://www.microsoft.com/typography/otspec/>*



**Adobe**