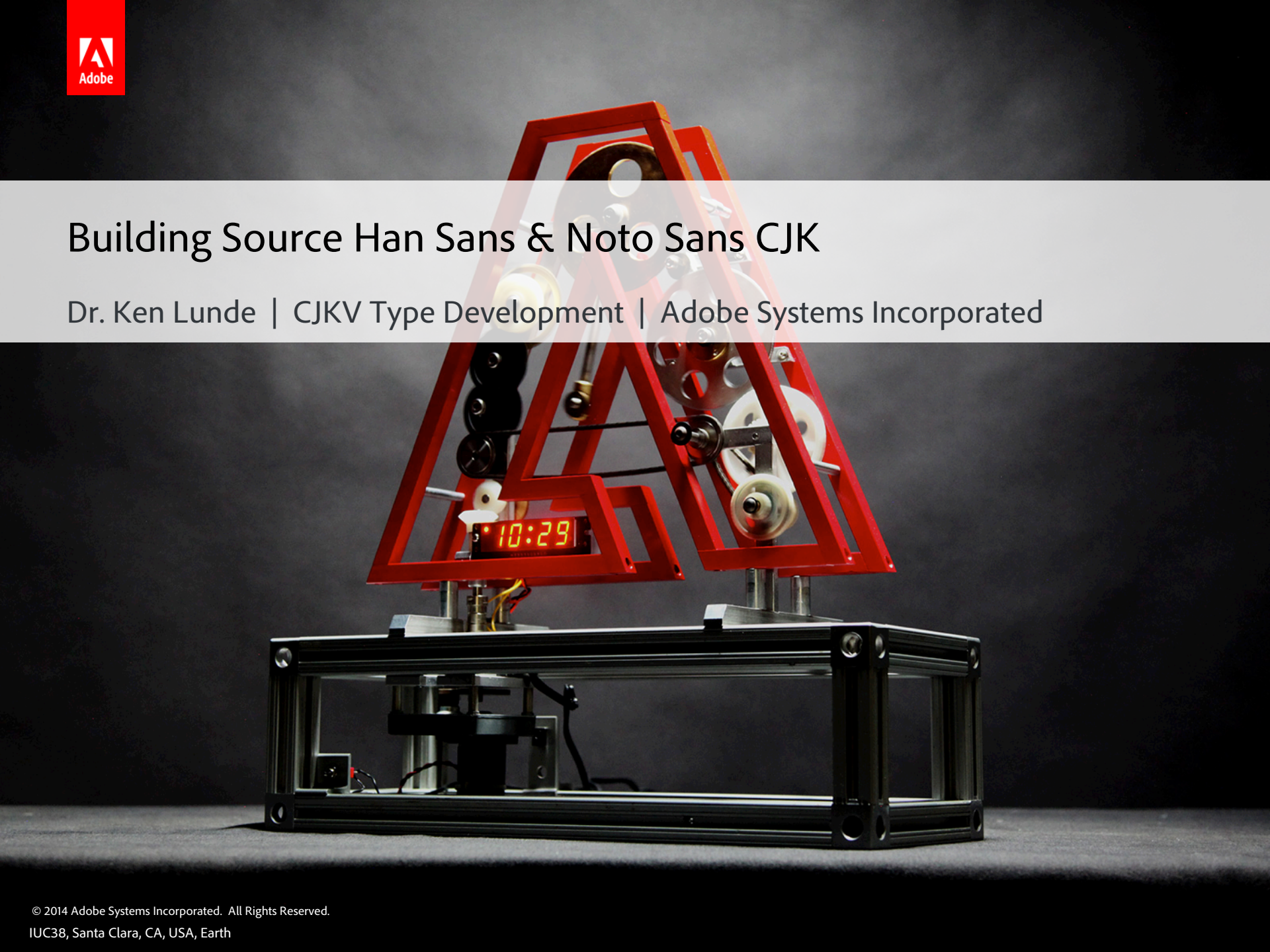
Building Source Han Sans & Noto Sans CJK

Dr. Ken Lunde | CJKV Type Development | Adobe Systems Incorporated



IUC38, Santa Clara, CA, USA, Earth

In The Beginning...

In The Beginning...

In The Beginning…

- There were no glyphs…

In The Beginning…

- There were no glyphs…

  …because Adobe and Google lawyers were duking it out

In The Beginning…

- There were no glyphs…

  …because Adobe and Google lawyers were duking it out

In The Beginning…

- There were no glyphs…

  …because Adobe and Google lawyers were duking it out

- Fast forward to 2014, there were now 65,535 glyphs in seven weights

In The Beginning…

- There were no glyphs…

  …because Adobe and Google lawyers were duking it out

- Fast forward to 2014, there were now 65,535 glyphs in seven weights
- How does one process 458,745 glyphs to become installable font resources?

In The Beginning…

- There were no glyphs…

  …because Adobe and Google lawyers were duking it out
- Fast forward to 2014, there were now 65,535 glyphs in seven weights
- How does one process 458,745 glyphs to become installable font resources?
  - HINT: Using unique Unicode-based working glyph names helps a lot!

In The Beginning…

- There were no glyphs…

  …because Adobe and Google lawyers were duking it out
- Fast forward to 2014, there were now 65,535 glyphs in seven weights
- How does one process 458,745 glyphs to become installable font resources?
  - HINT: Using unique Unicode-based working glyph names helps a lot!
- **8 easy steps!** ☺

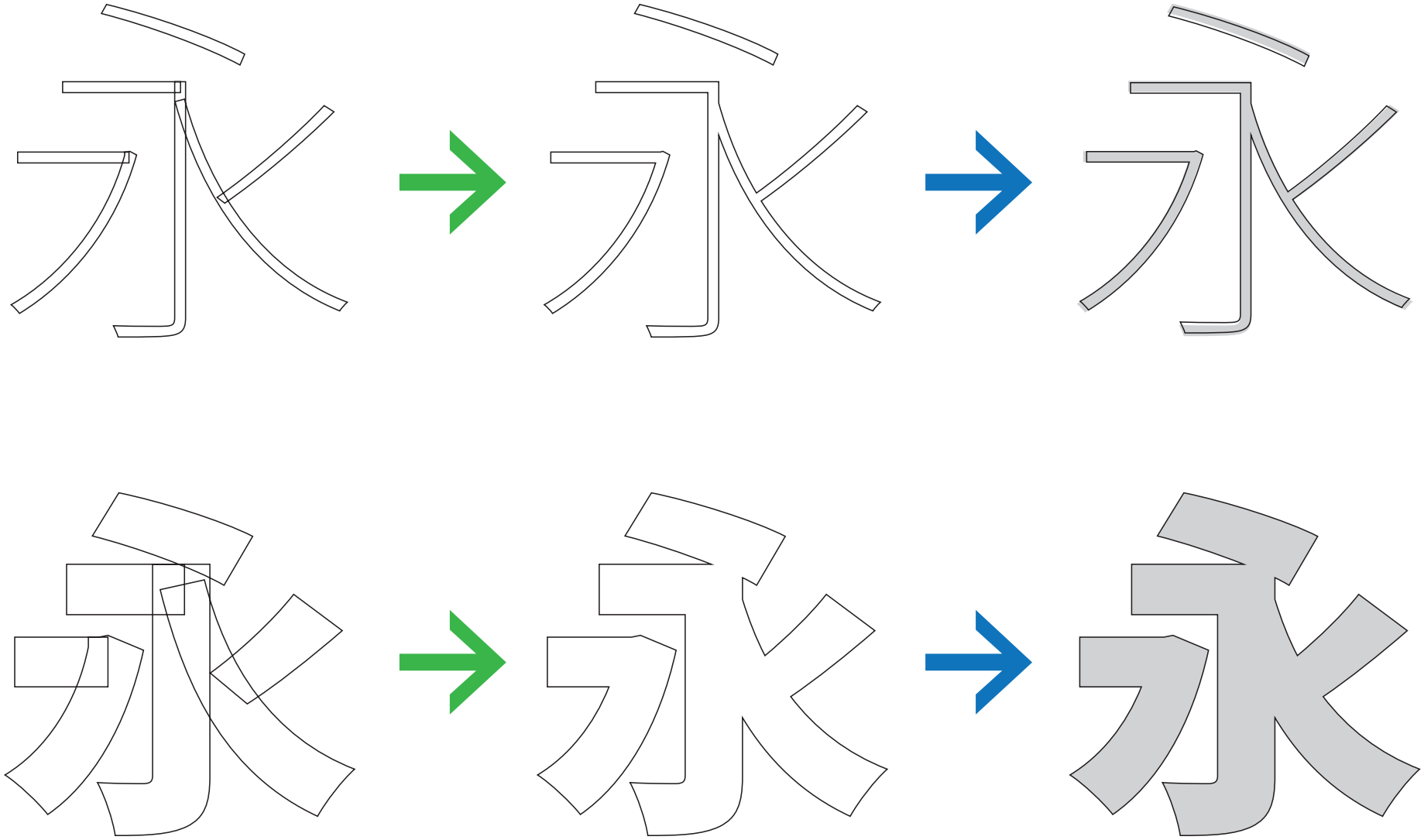
Step 1: Process Each Type Foundry's Glyph Data

- Adobe-Japan1-6 glyphs are completely processed by Adobe's team in Japan
  - Glyphs outside of Adobe-Japan1-6 are partially processed by Adobe's team in Japan
  - Receive data in three formats: *CIDFont resources, name-keyed fonts* & *UFOs*
  - Build two CIDFont resources per weight: *Adobe-Japan1-6* & *non–Adobe-Japan1-6*

Step 1: Process Each Type Foundry's Glyph Data

- Adobe-Japan1-6 glyphs are completely processed by Adobe's team in Japan
  - Glyphs outside of Adobe-Japan1-6 are partially processed by Adobe's team in Japan
  - Receive data in three formats: *CIDFont resources, name-keyed fonts* & *UFOs*
  - Build two CIDFont resources per weight: *Adobe-Japan1-6* & *non–Adobe-Japan1-6*
- TWB2 glyph data from Changzhou SinoType, Iwata & Sandoll
  - Generate row fonts for all seven weights—*using TWB2 (Type Work Bench 2)*
  - Build unhinted CIDFont resources
  - Remove overlapping subpaths using the AFDKO *checkOutlines* tool
  - Apply script-based non-linear scaling and script-based baseline shifting
    - Using the AFDKO *IS* (Intelligent Scaling) and *rotateFont* tools
  - Build four CIDFont resources per weight: *Changzhou SinoType (2), Iwata* & *Sandoll*

Overlapping Subpath Removal & Non-Linear Scaling: uni6C38-CN



Step 2: Assemble Interim 65,535-Glyph CIDFont Resources

- Create CID-based mappings for the seven source CIDFont resources

Step 2: Assemble Interim 65,535-Glyph CIDFont Resources

- Create CID-based mappings for the seven source CIDFont resources
- Use the AFDKO *mergeFonts* tool to assemble 65,535-glyph interim fonts

Type Foundry	CID Range	Glyphs	Content
Adobe	0–14453	14,454	Adobe-Japan1-6 (subset)
	14454–15924	1,471	Outside Adobe-Japan1-6
Iwata	15925–17464	1,540	JP ideographs
	17465–17626	162	KR ideographs
Changzhou SinoType	17627–24198	6,572	CN non-URO
	24199–41980	17,782	CN URO
	41981–42200	220	CN non-URO
	42201–42344	144	TW non-URO
	42345–48676	6,332	TW URO
	48677–48692	16	TW non-URO
	48693–49004	312	HK non-URO
	49005–50053	1,049	HK URO
	50054–51760	1,707	HK non-URO
Sandoll	51761–65484	13,724	Hangul
N/A	65485–65534	50	Reserved

Unused Glyphs

- Not all glyphs that were supplied were used, or could be used

Unused Glyphs

- Not all glyphs that were supplied were used, or could be used
- 1,517 Adobe-designed JP glyphs (Adobe-Japan1-6 kanji) were removed
  - Replaced by SinoType-designed CN glyphs that were deemed identical

Unused Glyphs

- Not all glyphs that were supplied were used, or could be used
- 1,517 Adobe-designed JP glyphs (Adobe-Japan1-6 kanji) were removed
  - Replaced by SinoType-designed CN glyphs that were deemed identical
- 1,032 Iwata-designed JP & KR glyphs were removed
  - Replaced by SinoType-designed CN glyphs that were deemed identical

Unused Glyphs

- Not all glyphs that were supplied were used, or could be used
- 1,517 Adobe-designed JP glyphs (Adobe-Japan1-6 kanji) were removed
  - Replaced by SinoType-designed CN glyphs that were deemed identical
- 1,032 Iwata-designed JP & KR glyphs were removed
  - Replaced by SinoType-designed CN glyphs that were deemed identical
- 3,458 additional Iwata-designed JP glyphs were removed
  - To make room for SinoType-designed TW & HK glyphs that had higher priority

Step 3: Decompile & Rebuild CIDFont Resources With Final Ordering

- Decompile 65,535-glyph CIDFont resources into 409 name-keyed fonts
  - Using the *decid* tool (not included in AFDKO) and the **interim** ordering file
  - A two-level directory hierarchy is created: *hint directory* & *rowfont directory*

Step 3: Decompile & Rebuild CIDFont Resources With Final Ordering

- Decompile 65,535-glyph CIDFont resources into 409 name-keyed fonts
  - Using the *decid* tool (not included in AFDKO) and the interim ordering file
  - A two-level directory hierarchy is created: *hint directory* & *rowfont directory*
- Rebuild the 65,535-glyph CIDFont resources
  - Create "cidfontinfo" files
  - Using the *makeCIDFont* tool (not included in AFDKO) and the **final** ordering file

Step 3: Decompile & Rebuild CIDFont Resources With Final Ordering

- Decompile 65,535-glyph CIDFont resources into 409 name-keyed fonts
  - Using the *decid* tool (not included in AFDKO) and the interim ordering file
  - A two-level directory hierarchy is created: *hint directory* & *rowfont directory*
- Rebuild the 65,535-glyph CIDFont resources
  - Create "cidfontinfo" files
  - Using the *makeCIDFont* tool (not included in AFDKO) and the final ordering file
- **The interim and final ordering files differ only in the order of the glyphs**

Step 3: Decompile & Rebuild CIDFont Resources With Final Ordering

- Decompile 65,535-glyph CIDFont resources into 409 name-keyed fonts
  - Using the *decid* tool (not included in AFDKO) and the interim ordering file
  - A two-level directory hierarchy is created: *hint directory* & *rowfont directory*
- Rebuild the 65,535-glyph CIDFont resources
  - Create "cidfontinfo" files
  - Using the *makeCIDFont* tool (not included in AFDKO) and the final ordering file
- **The interim and final ordering files differ only in the order of the glyphs**
- Establish hinting parameters and alignment zones
  - Using the AFDKO *stemHist* tool

Step 3: Decompile & Rebuild CIDFont Resources With Final Ordering

- Decompile 65,535-glyph CIDFont resources into 409 name-keyed fonts
  - Using the *decid* tool (not included in AFDKO) and the interim ordering file
  - A two-level directory hierarchy is created: *hint directory* & *rowfont directory*
- Rebuild the 65,535-glyph CIDFont resources
  - Create "cidfontinfo" files
  - Using the *makeCIDFont* tool (not included in AFDKO) and the final ordering file
- **The interim and final ordering files differ only in the order of the glyphs**
- Establish hinting parameters and alignment zones
  - Using the AFDKO *stemHist* tool
- Hint the glyphs using the AFDKO *autohint* tool

Final Glyph Ordering

- The interim glyph ordering keeps glyphs for each type foundry separate

Final Glyph Ordering

- The interim glyph ordering keeps glyphs for each type foundry separate
- The final glyph ordering uses Unicode-based glyph ordering

Final Glyph Ordering

- The interim glyph ordering keeps glyphs for each type foundry separate
- The final glyph ordering uses Unicode-based glyph ordering
  - Encoded singletons & region-specific ideographs: *-JP, -KR, -CN, -TW, -HK*
    - CIDs 1 through 61768 (61,768 glyphs)

Final Glyph Ordering

- The interim glyph ordering keeps glyphs for each type foundry separate
- The final glyph ordering uses Unicode-based glyph ordering
  - Encoded singletons & region-specific ideographs: *-JP, -KR, -CN, -TW, -HK*
    - CIDs 1 through 61768 (61,768 glyphs)
  - Unicode sequences, JIS90 (JP-specific) ideographs & *Adobe-Japan1* IVSes
    - CIDs 61769 through 62994 (1,226 glyphs)

Final Glyph Ordering

- The interim glyph ordering keeps glyphs for each type foundry separate
- The final glyph ordering uses Unicode-based glyph ordering
  - Encoded singletons & region-specific ideographs: *-JP, -KR, -CN, -TW, -HK*
    - CIDs 1 through 61768 (61,768 glyphs)
  - Unicode sequences, JIS90 (JP-specific) ideographs & *Adobe-Japan1* IVSes
    - CIDs 61769 through 62994 (1,226 glyphs)
  - Region-specific punctuation & proportional/half-width/full-width glyphs
    - CIDs 62995 through 63156 (162 glyphs)

Final Glyph Ordering

- The interim glyph ordering keeps glyphs for each type foundry separate
- The final glyph ordering uses Unicode-based glyph ordering
  - Encoded singletons & region-specific ideographs: *-JP, -KR, -CN, -TW, -HK*
    - CIDs 1 through 61768 (61,768 glyphs)
  - Unicode sequences, JIS90 (JP-specific) ideographs & *Adobe-Japan1* IVSes
    - CIDs 61769 through 62994 (1,226 glyphs)
  - Region-specific punctuation & proportional/half-width/full-width glyphs
    - CIDs 62995 through 63156 (162 glyphs)
  - Pre-composed old hangul syllables (500) & combining jamo
    - CIDs 63157 through 65144 (1,988 glyphs)

Final Glyph Ordering

- The interim glyph ordering keeps glyphs for each type foundry separate
- The final glyph ordering uses Unicode-based glyph ordering
  - Encoded singletons & region-specific ideographs: *-JP, -KR, -CN, -TW, -HK*
    - CIDs 1 through 61768 (61,768 glyphs)
  - Unicode sequences, JIS90 (JP-specific) ideographs & *Adobe-Japan1* IVSes
    - CIDs 61769 through 62994 (1,226 glyphs)
  - Region-specific punctuation & proportional/half-width/full-width glyphs
    - CIDs 62995 through 63156 (162 glyphs)
  - Pre-composed old hangul syllables (500) & combining jamo
    - CIDs 63157 through 65144 (1,988 glyphs)
  - Vertical glyphs
    - CIDs 65145 through 65484 (340 glyphs)

Step 4: Generate UTF-32 CMap Resources

- One per language, and built using Perl scripts that reference overrides
  - Simplified Chinese, Traditional Chinese (Taiwan), Japanese & Korean
  - Traditional Chinese (Hong Kong) forthcoming

Step 4: Generate UTF-32 CMap Resources

- One per language, and built using Perl scripts that reference overrides
  - Simplified Chinese, Traditional Chinese (Taiwan), Japanese & Korean
  - Traditional Chinese (Hong Kong) forthcoming
- The Japanese CMap resource is built first
  - Glyphs with working names that include "-JP" are preferred

Step 4: Generate UTF-32 CMap Resources

- One per language, and built using Perl scripts that reference overrides
  - Simplified Chinese, Traditional Chinese (Taiwan), Japanese & Korean
  - Traditional Chinese (Hong Kong) forthcoming
- The Japanese CMap resource is built first
  - Glyphs with working names that include "-JP" are preferred
- The Korean CMap resource is based on **Japanese**
  - Glyphs with working names that include "-KR" are preferred

Step 4: Generate UTF-32 CMap Resources

- One per language, and built using Perl scripts that reference overrides
  - Simplified Chinese, Traditional Chinese (Taiwan), Japanese & Korean
  - Traditional Chinese (Hong Kong) forthcoming
- The Japanese CMap resource is built first
  - Glyphs with working names that include "-JP" are preferred
- The Korean CMap resource is based on Japanese
  - Glyphs with working names that include "-KR" are preferred
- The Simplified Chinese CMap resource is based on **Japanese**
  - Glyphs with working names that include "-CN" are preferred

Step 4: Generate UTF-32 CMap Resources

- One per language, and built using Perl scripts that reference overrides
  - Simplified Chinese, Traditional Chinese (Taiwan), Japanese & Korean
  - Traditional Chinese (Hong Kong) forthcoming
- The Japanese CMap resource is built first
  - Glyphs with working names that include "-JP" are preferred
- The Korean CMap resource is based on Japanese
  - Glyphs with working names that include "-KR" are preferred
- The Simplified Chinese CMap resource is based on Japanese
  - Glyphs with working names that include "-CN" are preferred
- The Traditional Chinese CMap resource is based on **Simplified Chinese**
  - Glyphs with working names that include "-TW" or "-HK" are preferred

Step 4: Generate UTF-32 CMap Resources

- One per language, and built using Perl scripts that reference overrides
  - Simplified Chinese, Traditional Chinese (Taiwan), Japanese & Korean
  - Traditional Chinese (Hong Kong) forthcoming
- The Japanese CMap resource is built first
  - Glyphs with working names that include "-JP" are preferred
- The Korean CMap resource is based on Japanese
  - Glyphs with working names that include "-KR" are preferred
- The Simplified Chinese CMap resource is based on Japanese
  - Glyphs with working names that include "-CN" are preferred
- The Traditional Chinese CMap resource is based on Simplified Chinese
  - Glyphs with working names that include "-TW" or "-HK" are preferred
- Use the *cmap-tool.pl* tool to compile the raw mappings into CMap resources

Step 5: Generate UVS Definition Files

- Only for Traditional Chinese, Japanese & Korean, and built using Perl scripts
  - There are no registered IVSes nor Standardized Variants for Simplified Chinese

Step 5: Generate UVS Definition Files

- Only for Traditional Chinese, Japanese & Korean; built using Perl scripts
  - There are no registered IVSes nor Standardized Variants for Simplified Chinese
- *SourceHanSans_JP_sequences.txt*: *Adobe-Japan1* IVSes & Standardized Variants

Step 5: Generate UVS Definition Files

- Only for Traditional Chinese, Japanese & Korean; built using Perl scripts
  - There are no registered IVSes nor Standardized Variants for Simplified Chinese
- *SourceHanSans_JP_sequences.txt*: *Adobe-Japan1* IVSes & Standardized Variants
- *SourceHanSans_KR_sequences.txt*: Standardized Variants

Step 5: Generate UVS Definition Files

- Only for Traditional Chinese, Japanese & Korean; built using Perl scripts
  - There are no registered IVSes nor Standardized Variants for Simplified Chinese
- *SourceHanSans_JP_sequences.txt*: *Adobe-Japan1* IVSes & Standardized Variants
- *SourceHanSans_KR_sequences.txt*: Standardized Variants
- *SourceHanSans_TWHK_sequences.txt*: Standardized Variants

Step 5: Generate UVS Definition Files

- Only for Traditional Chinese, Japanese & Korean; built using Perl scripts
  - There are no registered IVSes nor Standardized Variants for Simplified Chinese
- *SourceHanSans_JP_sequences.txt*: *Adobe-Japan1* IVSes & Standardized Variants
- *SourceHanSans_KR_sequences.txt*: Standardized Variants
- *SourceHanSans_TWHK_sequences.txt*: Standardized Variants
- These files are used to build Format 14 (UVS) 'cmap' subtables
  - UVS = *Unicode Variation Sequence* (registered IVSes & Standardized Variants)

Step 6: Generate AFDKO “features” Files

- An AFDKO “features” file specifies GPOS/GSUB features and table overrides

Step 6: Generate AFDKO “features” Files

- An AFDKO “features” file specifies GPOS/GSUB features and table overrides
- Raw “features” file data uses working glyph names
  - Working glyph names are converted to CIDs when compiling final “features” files
  - This insulates against CID changes between interim (and future) versions of the fonts

Step 7: Create AFDKO "FontMenuNameDB" Files

- An AFDKO "FontMenuNameDB" file specifies menu names
  - English menu names are required
  - Localized menu names are optional

Step 7: Create AFDKO "FontMenuNameDB" Files

- An AFDKO "FontMenuNameDB" file specifies menu names
  - English menu names are required
  - Localized menu names are optional

```
[SourceHanSans-ExtraLight]
     f=3,1,0x411,\6E90\30CE\89D2\30B4\30B7\30C3\30AF
     s=3,1,0x411,ExtraLight
     l=3,1,0x411,\6E90\30CE\89D2\30B4\30B7\30C3\30AF ExtraLight
     f=Source Han Sans
     s=ExtraLight
     l=Source Han Sans ExtraLight

[NotoSansCJKjp-Thin]
     f=Noto Sans CJK JP
     s=Thin
     l=Noto Sans CJK JP Thin
```

Step 8: Use AFDKO *makeotf* & *otf2otc* Tools To Build OTFs & OTCs

- Command lines are provided in the "COMMANDS.txt" file

Step 8: Use AFDKO *makeotf* & *otf2otc* Tools To Build OTFs & OTCs

- Command lines are provided in the "COMMANDS.txt" file
- Required files for the AFDKO *makeotf* tool
  - "cidfontinfo" file
  - CIDFont resource
  - UTF-32 CMap resource
  - UVS definition file—*optional*
  - "features" file
  - "FontMenuNameDB" file

Step 8: Use AFDKO *makeotf* & *otf2otc* Tools To Build OTFs & OTCs

- Command lines are provided in the "COMMANDS.txt" file
- Required files for the AFDKO *makeotf* tool
  - "cidfontinfo" file
  - CIDFont resource
  - UTF-32 CMap resource
  - UVS definition file—*optional*
  - "features" file
  - "FontMenuNameDB" file
- Required files for the AFDKO *otf2otc* tool
  - Two or more OTFs

Source Han Sans & Noto Sans CJK: Seven Weights & Four Languages!

思源黑体 思源黑體 源ノ角ゴシック 본고딕

思源黑体 思源黑體 源ノ角ゴシック 본고딕

思源黑体 思源黑體 源ノ角ゴシック 본고딕

思源黑体 思源黑體 源ノ角ゴシック 본고딕

思源黑体 思源黑體 源ノ角ゴシック 본고딕

思源黑体 思源黑體 源ノ角ゴシック 본고딕

思源黑体 思源黑體 源ノ角ゴシック 본고딕

Procurement Portals

- Source Han Sans
  - Installable fonts → *https://github.com/adobe-fonts/source-han-sans/releases/*
  - Typekit Desktop Sync → *http://www.typekit.com/*
  - Sources → *https://github.com/adobe-fonts/source-han-sans/*
    - AFDKO (OS X, Windows & Linux) → *http://www.adobe.com/devnet/opentype/afdko.html*
    - AFDKO "open source" sources → *https://github.com/adobe-type-tools/afdko/*
- Noto Sans CJK
  - Installable fonts → *http://www.google.com/get/noto/*

Feedback Is Welcomed & Encouraged!

- Source Han Sans

  https://github.com/adobe-fonts/source-han-sans/issues/
- Noto Sans CJK

  https://code.google.com/p/noto/issues/

More Information

- CJK Type Blog

  http://blogs.adobe.com/CCJKType/

- Official ReadMe

  https://github.com/adobe-fonts/source-han-sans/raw/release/SourceHanSansReadMe.pdf

A Font Solution For More Than 1.5 Billion People