INTERNATIONAL ORGANISATION FOR STANDARDISATION

ORGANISATION INTERNATIONALE DE NORMALISATION

ISO/IEC JTC 1/SC 29/WG 3

CODING OF MOVING PICTURES AND AUDIO

ISO/IEC JTC 1/SC 29/WG 3 m NNNN

Mirkwood Service Station – March 2024

Title: Updates based on Public Feedback to Changes to the OFF Font Format

Author: Dave Crossland (Google Inc., dcrossland@google.com), Behdad Esfahbod (behdad@behdad.org), Laurence Penney (lorp@lorp.org), Liam Quin (Delightful Computing, liam@delightfulcomputing.com), Rod Sheeter (Google Inc., rsheeter@google.com)

Introduction

This introduction is to give context and is not itself proposed text.

A recent proposal to the ISO MPEG OpenFont committee, m66260, was accepted in January of 2024. Since that time there have been a number of comments on the proposal.

Some of these pointed out small typographical errors, and these are included in this Proposal.

Some of the comments made technical suggestions, and, where appropriate, these are also incorporated in this proposal.

The primary goal is to enable font producers to create fonts containing more than 65535 glyphs, and to guide implementors, technical writers, trainers, and others, in their use.

Technical discussion of most the items proposed here may be found in Github issues, as noted for each change.

hdmx

https://github.com/harfbuzz/boring-expansion-spec/issues/131

In 5.6.2 hdmx—Horizontal device metrics, in the Device Record table headed Each DeviceRecord for format 0 looks like this, change the table as follows:

Each DeviceRecord for format 0 looks like this.

Device Record		
Type	**Name**	**Description**
uint8	pixelSize	Pixel size for following widths (as ppem).
uint8	maxWidth	Maximum width.
uint8	widths[numGlyphs]	Array of widths (numGlyphs is from the 'MAXP' table if present, otherwise ‘maxp’).

LTSH

https://github.com/harfbuzz/boring-expansion-spec/issues/132

In 5.6.4 LTSH—Linear threshold, at the end of the section, add Format 1 as follows:

Format 1 of the ‘LTSH’ table supports more than 65535 glyphs in a font:

Type	Name	Description
uint16	version	Version number (set to 1).

uint24	numGlyphs	Number of glyphs (numGlyphs is from the 'MAXP' table if present, otherwise ‘maxp’).
uint8	yPels[numGlyphs]	The vertical pel height at which the glyph can be assumed to scale linearly. On a per glyph basis.

In 6.3.5.1 JSTF—The justification table, after JsfScriptRecord and before Justification script table, insert the following new subsection, just after “Example 1 at the end of this clause shows a JSTF Header table and JstfScriptRecord.”

JSTF header 1.1

Type	Name	Description
uint16	majorVersion	Major version of the JSTF table, = 1
uint16	minorVersion	Minor version of the JSTF table, = 1
uint16	jstfScriptCount	Number of JstfScriptRecords in this table
JstfScriptRecord	jstfScriptRecords[jstfScriptCount]	Array of JstfScriptRecords, in alphabetical order by jstfScriptTag
uint16	jstfScriptCount2	Number of JstfScriptRecords2 in this table
JstfScriptRecord2	jstfScriptRecords2[jstfScriptCount]	Array of JstfScriptRecords2, in alphabetical order by jstfScriptTag

JstfScriptRecord2

Type	Name	Description
Tag	jstfScriptTag	4-byte JstfScript identification
Offset32	jstfScriptOffset	Offset to JstfScript2 table, from beginning of JSTF Header

After the JstfScript table, add:

The JstfScript2 table is based on the JstfScript table, but has 32-bit offsets:

JstfScript2 table

Type	Name	Description
Offset32	extenderGlyphOffset	Offset to ExtenderGlyph table, from beginning of JstfScript table (may be NULL)
Offset32	defJstfLangSysOffset	Offset to Default JstfLangSys table, from beginning of JstfScript2 table (may be NULL)
uint16	jstfLangSysCount	Number of JstfLangSysRecords in this table, may be zero (0)
JstfLangSysRecord	jstfLangSysRecords [jstfLangSysCount]	Array of JstfLangSysRecords, in alphabetical order by jstfLangSysTag

After the Extender Glyph table, just before Justification Language System table, insert:

ExtenderGlyph2 table

The *ExtenderGlyph2* table supports fonts containing more than 65535 glyphs.

Type	Name	Description
uint16	glyphCount	Number of extender glyphs in this script
uint24	extenderGlyphs[glyphCount]	Extender glyph IDs – in increasing numerical order

In the MultiItemVariationData table

https://github.com/harfbuzz/boring-expansion-spec/commit/3a911400bdd0ab5b9f1816a542a6202d048394f0

Change the format from uint16 to uint8.

Add a note immediately after the table:

NOTE 1 The format is encoded as an 8-bit value to save space.

In the Variable Composite Description, add a row to Variable Component Record, between glyphID and axisIndicesIndex:

uint32var	conditionSetIndex	Optional, only present if HAVE_CONDITION bit of flags is

set.

Add another row at the end of the same table (Variable Component Record), after FWORD TCentery:

uint32var reserved[] Optional: Process and discard one uint32var per each set bit in RESERVED_MASK

After the list of flags and before Variable Component Flags, add,

This specification does not define a meaning for any bits in RESERVED_MASK, and conforming fonts shall therefore set all such bits to zero. Font processing software that encounters bits in RESERVED_MASK that are set to one shall process them in turn by reading uint32var values, one for each set bit, for extension purposes.

Again in the Variable Composite Description, in Variable Component Flags, rename item 7 from USE_MY_METRICS to HAVE_CONDITION, and change the last entry (Reserved):

6 HAVE_CONDITION

. . .

15-31 RESERVED_MASK

To the VARC table header, add a new row below varStore and above axisIndicesList (not to be confused with axisIndicesIndex!)

Offset32 conditionSetList Offset to ConditionSetList, from the start of VAC table header.

Just before Processing of Variable Composite Glyphs, add the new ConditionSetList type:

ConditionSetList table

Type	Name	Description
Offset32	ConditionSet[]	Array of offsets from the beginning of the ConditionSetList table

After the first paragraph of Processing of Variable Composite Glyphs (The component glyphs to be loaded...) insert the following (just before the paragraph "For any unspecified axis")

For each parsed component, if the HAVE_CONDITION flag is set, that component shall be loaded but not used (for example, not displayed), unless the referenced ConditionSet evaluates to true. The referenced ConditionSet is found using conditionSetIndex and consulting the top-level condisionSetList.

NOTE: dmap and fvar changes were moved to separate documents.

[end]