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 m70320

Kemer, Turkey – November 2024

Title: Summary of proposed editorial and technical corrections in 14496-22/CD

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

1. Introduction

This document summarizes comments sent via various national bodies as comments from the Google Fonts team. The comments were sent separately because of differing deadlines.

2. Comments on Section 6

2.3. SequenceContextFormat4 table (technical, editorial)

In 6.2.8 SequenceContextFormat4 table, there could be overflow.

Make seqRuleSetCount a uint24 instead of uint16.

In addition, there is a minor editorial error: seqRuleSetOffsets should point to SequenceRuleSet2 tables, not SequenceRuleSet tables. The following table must be renamed to SequenceRuleSet2 table to avoid conflict with a table with the same name in format 1.

2.4. Sequence context format 5 (editorial)

In 6.2.8, the definition of ClassSequenceRule here is redundant; it has already been defined.

Remove the redundant table.

2.5. Chained Sequence Context (technical)

In 6.2.8, in ChainedSequenceContextFormat4, there could be overflow.

Make chainedSeqRuleSetCount be uint24 instead of uint16.

In chainedSequenceRuleSet2 table, chainedSeqRuleCount can be a uint16 instead of uint24. However, in ChainedSequenceRule2 table, lookaheadSequence must be uint24, not uint 16.

(this affects only the leftmost column of the table, not whether e.g. items are an array or not)

2.6. ChainedSequenceRule2 Spurious (technical)

In 6.2.8, ChainedSequenceRule2 table is not needed, because fonts are not likely to need (and cannot use) more than 65535 class values anywhere.

Remove the table, and ChainedSequenceContextFormat5 should use ChainedClassSequenceRule instead of ChainedClassSequenceRule2.

2.7. MarkGlyphSetsTable (technical)

In 6.3.2.2.6, we should consider adding a MarkGlyphSets format 2 table with a uint32 markGlyphSetCount, to allow more than 65535 different mark glyph sets.

Add a MarkGlyphSets format 2 table with a uint32 markGlyphSetCount, and the explanation,

Format 2 allows more than 65535 different mark glyph sets.+

2.8. PairSet2 table – PairValue Overflow (technical)

In 6.3.4.3.2 Lookup Type 2 subtable: pair adjustment positioning,
the PairSet2 table needs to refer to PairValue2 records, not PairValue recoreds, in order to support 24-bit glyph IDs.

Change PairValue to PairValue2 in the PairSet2 table; add a PairValue2 Record beneath it which is the same as PairValue but uses uint24 for the type of secondGlyph. This new table should replace the text, “See format 1 for PairValue record description.”

2.9. LigCaretList table – overflow

In 6.3.2.2.4 Ligature caret list table, fonts with many glyphs will overflow the 16-bit coverageOffset in this table, even if they do not have many ligatures.

In Example 4 LigCaretList table, make ligCaretListOffset2 point to a LigCaretList2 instead of current “ligature caret list table”

After the LigCaretList table, define a LigCaretList2 table that is the same as LigCaretList, but uses Offset32 coverageOffset, uint24 ligGlyphCount, and Offset24 ligGlyphOffsets[ligGlyphCount]

3. Comments on Section 7

3.3. SparseVariationRegionList (technical)

In section 7.2.3.5: In a large font there could be more than 65535 regions, so regionCount here should be uint32

Change regionCount to be uint32

3.4. MultiItemVariationData subtable (technical)

In 7.2.3.5. Implementation measurements have shown that this table is slow to parse. We add a Format 2 that is the same except for using float32 (to be defined) makes it much faster.

Add a format 2 for MultiItemVariationData subtable in which deltaSets is an Offset32 (as in format 1) but to a CFF2-style INDEX array of little-endian Float32, as defined in IEEE 754.

3.5. Processing of Variable Composite Glpyhs (editorial)

Subsection 7.3.10.5 Processing of Variable Composite Glyphs should come before 7.3.10.4 Hinting of Variable Composite Glyphs, as you have to know how to process them before the hinting section makes sense.

Suggest: swap sections 7.3.10.4 and 7.3.10.5

4. Comment on Section 4.3

4.3. The uint32var format (technical)

This variable-byte encoding is very efficient, but has the possibility of encoding a 36-bit number, which would overflow. A malicious font might make use of this.

Change the last entry of the table explaining uint32var from:

Four bytes follow; the low 4 bits of the first byte become the most significant bits of the new value, as above

to:

Four bytes follow; the low 4 bits of the first byte become the most significant bits of the new value, as above. Note that the least significant 4 bits of the initial byte must be zero in this case to prevent overflow.

5. Note on comment regarding VARC

We have decided to postpone the changes to VARC for a future amendment or revision; the current version works, with the minor errors in this document corrected, but research suggests further improvements may be possible.