 ISO/IEC JTC 1/SC 29/WG 3 N0056

**ISO/IEC JTC 1/SC 29/WG 3  
MPEG Systems   
Convenorship: KATS (Korea, Republic of)**

**Document type:** Output Document

**Title:** **Working Draft of ISO/IEC 14496-22:2019 AMD 2 Extending color font functionality and other updates**

**Status:** Approved

**Date of document:** 2020-10-23

**Source:** ISO/IEC JTC 1/SC 29/WG 3

# Expected action: None

# Action due date: None

**No. of pages:** 24 (with cover page)

**Email of Convenor:** young.L @ samsung . com

**Committee URL:** <https://isotc.iso.org/livelink/livelink/open/jtc1sc29wg3>

**INTERNATIONAL ORGANIZATION FOR STANDARDIZATION**

**ORGANISATION INTERNATIONALE DE NORMALISATION**

**ISO/IEC JTC 1/SC 29/WG 3 MPEG SYSTEMS**

**ISO/IEC JTC 1/SC 29/WG 3 N** **0056**

**October 2020, Virtual**

|  |  |
| --- | --- |
| **Title** | **Working Draft of ISO/IEC 14496-22:2019 AMD 2 Extending color font functionality and other updates** |
| **Source** | **WG 3, MPEG Systems** |
| **Status** | **Approved** |
| **Serial Number** | **19754** |

**ISO/IEC 14496-22:2019(4th edition)\_AMD2**

ISO JTC1/SC 29/WG 3

Secretariat: XXXX

**Information technology — Coding of audio-visual objects —**

Part 22: **Open Font Format**

WD stage

**Warning for WDs and CDs**

This document is not an ISO International Standard. It is distributed for review and comment. It is subject to change without notice and may not be referred to as an International Standard.

Recipients of this draft are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

© ISO 2020

All rights reserved. Unless otherwise specified, or required in the context of its implementation, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO’s member body in the country of the requester.

ISO copyright office

CP 401 • Ch. de Blandonnet 8

CH-1214 Vernier, Geneva

Phone: +41 22 749 01 11

Email: copyright@iso.org

Website: www.iso.org

Published in Switzerland

Contents

[Expected action: None 1](#_Toc54616357)

[Action due date: None 1](#_Toc54616358)

[Foreword vi](#_Toc54616359)

[Introduction vii](#_Toc54616360)

[1 Scope 1](#_Toc54616361)

[2 Amended content 1](#_Toc54616362)

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of ISO documents should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see [www.iso.org/directives](https://www.iso.org/directives-and-policies.html)).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see [www.iso.org/patents](https://www.iso.org/iso-standards-and-patents.html)).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT), see [www.iso.org/iso/foreword.html](https://www.iso.org/foreword-supplementary-information.html).

This document was prepared by Joint Technical Committee 1 ISO/IEC JTC1, Subcommittee SC 29, *[name of subcommittee]*.

This amendment replaces / updates certain clauses of the 4th edition (ISO/IEC 14496-22:2019), which has been technically revised.

The main changes compared to the previous edition are as follows:

— Updates to COLR table formats.

A list of all parts in the ISO/IEC 14496 series can be found on the ISO website.

Any feedback or questions on this document should be directed to the user’s national standards body. A complete listing of these bodies can be found at [www.iso.org/members.html](https://www.iso.org/members.html).

Introduction

The International Organization for Standardization (ISO) *[and/or] International Electrotechnical Commission (IEC)* draw[s] attention to the fact that it is claimed that compliance with this document may involve the use of a patent.

ISO *[and/or] IEC* take[s] no position concerning the evidence, validity and scope of this patent right.

The holder of this patent right has assured ISO *[and/or] IEC* that he/she is willing to negotiate licences under reasonable and non-discriminatory terms and conditions with applicants throughout the world. In this respect, the statement of the holder of this patent right is registered with ISO *[and/or] IEC*. Information may be obtained from the patent database available at [www.iso.org/patents](http://www.iso.org/patents).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights other than those in the patent database. ISO *[and/or] IEC* shall not be held responsible for identifying any or all such patent rights.

**Information technology — Coding of audio-visual objects —** Part 22: **Open Font Format**

# Scope

This amendment extends the color font capabilities to implement feature-rich variable color fonts with multiple layers supporting complex design using advanced graphical primitives. It also introduces technical changes, technical clarifications and updates, to other parts of the standard.

# Amended content

4.3

*Replace the table defining data types with the following:*

|  |  |
| --- | --- |
| **Data Type** | **Description** |
| uint8 | 8-bit unsigned integer. |
| int8 | 8-bit signed integer. |
| uint16 | 16-bit unsigned integer. |
| int16 | 16-bit signed integer. |
| uint24 | 24-bit unsigned integer. |
| uint32 | 32-bit unsigned integer. |
| int32 | 32-bit signed integer. |
| Fixed | 32-bit signed fixed-point number (16.16) |
| FWORD | int16 that describes a quantity in font design units. |
| UFWORD | uint16 that describes a quantity in font design units. |
| F2DOT14 | 16-bit signed fixed number with the low 14 bits of fraction (2.14). |
| LONGDATETIME | Date represented in number of seconds since 12:00 midnight, January 1, 1904. The value is represented as a signed 64-bit integer. |
| Tag | Array of four uint8s (length = 32 bits) used to identify a table, design-variation axis, script, language system, feature, or baseline |
| Offset16 | Short offset to a table, same as uint16, NULL offset = 0x0000 |
| Offset24 | 24-bit offset to a table, same as uint24, NULL offset = 0x000000 |
| Offset32 | Long offset to a table, same as uint32, NULL offset = 0x00000000 |

5.7.11

*Replace the first four paragraphs with the following:*

The COLR table adds support for multi-colored glyphs in a manner that is compatible with existing text engines and relatively easy to support with current OFF font files.

The COLR table defines a list of base glyphs, which are typically regular glyphs, often associated with a ‘cmap’ entry. Each base glyph is associated with a set of glyphs composed together to create a colored presentation for the base glyph. The layered glyphs are explicitly defined in bottom-up z-order and each of their advance widths must match those of the base glyph. If the font has vertical metrics, the associated layer glyphs must also have the same advance height and vertical Y origin as the base glyph.

The COLR table works together with the CPAL table, which holds the color palettes used by the color composition. The COLR table has a dependency on the CPAL table. If the COLR table is present in a font but no CPAL table exists, then the COLR table is ignored.

*Add a new subclause 5.7.11.1 “Color table versions and design concepts” with the following text:*

Two versions of the COLR table are defined.

Version 0 allows for simple composition of colored elements: A linear sequence of glyphs that are stacked vertically (z-order) as layers. Each layer combines a glyph outline from the ‘glyf’, CFF, or CFF2 table (referenced by glyph ID) with a solid color fill.

Version 1 supports much richer capabilities:

* The colored presentation for a base glyph can use a directed acyclic graph of elements, with nodes in the graph corresponding to sub compositions that are vertically layered.
* The individual layers can be glyph outlines, as in version 0. But the can also be compositions of elements, including complete structure defined as the colored presentation for another base glyph.
* Fills are not limited to solid colors and can use different types of gradients (see subclause 5.7.11.2).
* Several composition and blending modes are supported, providing options for how elements are graphically composed.

In addition, a COLOR version 0 table can be used in variable fonts with glyph outlines being variable, but no other aspects of the color composition being variable. In COLR version 1, several additional items can be variable:

* The design grid coordinates used to define gradients.
* The elements in transformation matrices.
* The relative placement of gradient color stops on a color line.
* The alpha values applied to individual colors.

***Editor’s note:*** Additional contributions are needed to define the processing of COLR glyph sequences, graphic composition concepts, and variation mechanisms for newly added COLR v1 values.

*Add a new subclause 5.7.11.2 “Graphical primitives” with the following text:*

The two main graphical primitives that are defined for COLR version 1 table are linear gradients and radial gradients.

In most graphics systems, linear gradients are declared using two points, and radial gradients are declared using two circles. Such graphics systems also support a transformation matrix, via which one can get shear in linear gradients or arbitrary ellipses with radial gradients. Since our proposed format does *not* have such universal transform underneath, the definitions of linear and radial gradients are extended from their typical form to accommodate for these transformations in the gradient declaration itself.

**Color line**

A color line is a function that maps real numbers to a color value to define a 1-dimensional gradient, to be used and referenced from *Linear Gradients* and *Radial Gradients*. Colors of the gradient are defined by *color stops*.

Color stops are defined at color stop positions. Color stop position 0 maps to the start point of a linear gradient or the center of the first circle of a radial gradient. Color stop position 1 maps to the end point of a linear gradient or the center of the second circle of a radial gradient. In the interval [0, 1] the color line must contain at least one color stop, but may contain multiple color stops that define the gradient.

Outside the defined interval, the gradient pattern in between the outer defined positions is repeated according to the color line *extend mode*.

If there are multiple color stops defined for the same coordinate, the first one is used for computing the color value for values below the coordinate, the last one is used for computing the color value for values above. All other color stops for this coordinate are ignored.

Limiting the specified interval to a sub-range of [0, 1] allows for looping through colors repeatedly along the mapped distance, without having to encode them multiple times. In that sense, our color line is similar to CSS repeating-linear-gradient() and repeating-radial-gradient() functions [32].

In order to achieve:

* one gradient along the gradient positions (linear, or radial) and padded colors outside this range, color stops at 0 and 1 must be defined, and color line extend mode *pad* must be used. This achieves similarly behavior as defined in CSS linear-gradient() and radial-gradient() functions [32].
* a repeated gradient along the gradient positions (linear or radial): divide 1 by the number of desired repetitions and use the result as your maximum color stop, then use color line extend mode *repeat* to have it continue outside the defined interval.
* a mirrored / color-circle gradient: divide 1 by two times the number of desired full color stripes, and define the color stops between the 0 and the result of this division, then use color line extend mode *reflect* to have it continue mirrored.

**Extend Mode**

Three extend modes are defined to control the behavior of the gradient outside its specified endpoints:

*Extend Pad*

For numbers outside the defined interval the color line continues to map to the outer color values, i.e. for values less than the leftmost defined color stop, it maps to the leftmost color stop value; for values greater than the rightmost defined color stop value, it maps to the rightmost defined color value.

*Extend Repeat*

For numbers outside the interval, the color line continues to map as if the defined interval was repeated.

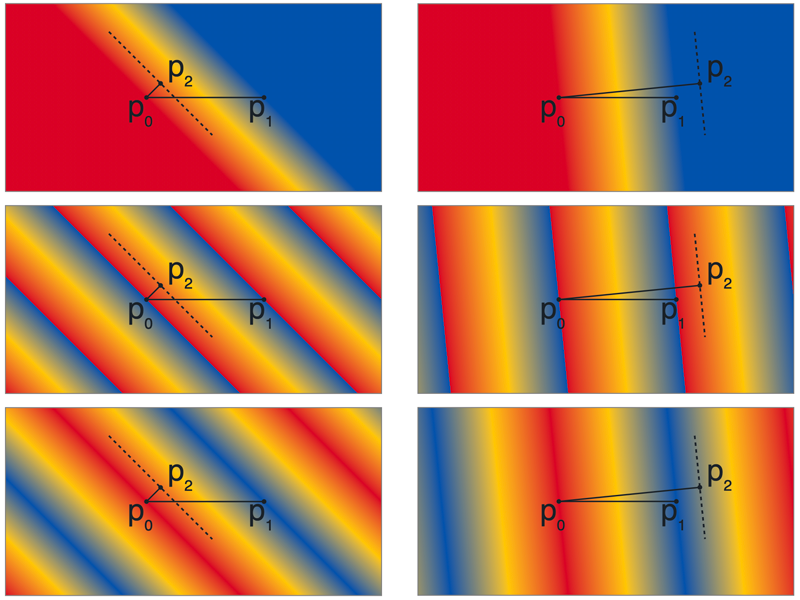
*Extend Reflect*

For numbers outside the defined interval, the color continues to map as if the interval would continue mirrored from the previous interval. This allows defining stripes in rotating colors.

**Linear Gradients**

Linear gradients are defined by two color line points P0 and P1, between which a gradient is interpolated. A point P2 is defined to rotate the gradient angle / orientation separately from the color line endpoints.

If the dot-product (P₁ - P₀) . (P₂ - P₀) is zero (or near-zero for an implementation-defined definition) then gradient is ill-formed and nothing must be rendered.



**Figure 5.6 – Examples of linear gradients and their defining points with extend modes pad, repeat and reflect (top to bottom) with color stops for blue at 0, yellow at 0.5 and red at 1.**

**Radial Gradients**

Radial gradients are defined based on circles. If subject to a transform (via PaintTransformed) those circles may become ellipses.

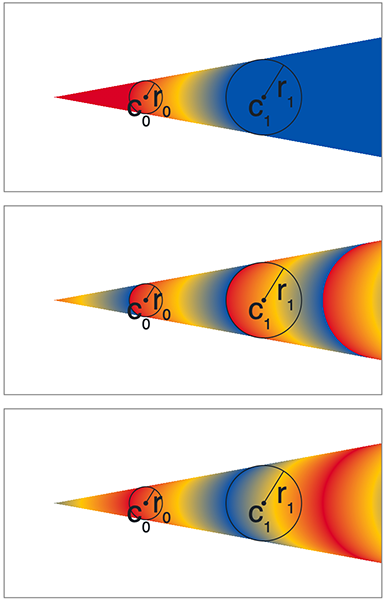
A radial gradient is defined as a gradient between two (optionally transformed) circles with center c0 and radius r0, and center c1 and radius r1, and a specified color line. The circle c0, r0 will be drawn with the color at color line position 0. The circle c1, r1 will be drawn with the color at color line position 1.

The drawing algorithm for radial gradients follows the HTML canvas specification [33], with circle center points c0 and c1 defined as c0 = (x0, y0) and c1 = (x1, y1).

***Editor’s note:*** Can the following description be replaced by an existing or new reference? Duplicating descriptions from another spec is generally frowned upon, the established practice is to provide a specific reference to the source where this is described in details.

Radial gradients are rendered by following these steps:

1. If c0 = c1 and r0 = r1 then the radial gradient must paint nothing.
2. Let x(ω) = (x1-x0)ω + x0 Let y(ω) = (y1-y0)ω + y0 Let r(ω) = (r1-r0)ω + r0  
   Let the color at ω be the color at that position on the gradient color line (with the colors coming from the interpolation and extrapolation described above).
3. For all values of ω where r(ω) > 0, starting with the value of ω nearest to positive infinity and ending with the value of ω nearest to negative infinity, draw the circumference of the ellipse resulting from translating circle with radius r(ω) by affine transform at position (x(ω), y(ω)), with the color at ω, but only painting on the parts of the bitmap that have not yet been painted on by earlier circles in this step for this rendering of the gradient.



**Figure 5.7 – Example of a radial gradient rendering with extend modes pad, repeat and reflect (top to bottom) with color stops for blue at 0, yellow at 0.5 and red at 1.**

NOTE: Implementations need to be careful to properly render radial gradient even if they are subject to a *degenerate* or *near-degenerate* transform. Such radial gradients do have a well-defined shape, which is a strip or a cone filled with a linear gradient.

*Add a new subclause 5.7.11.3 “Color table formats” with the following text:*

**Header**

The COLR table begins with a header. Two versions have been defined. Offsets in the header are from the start of the table.

*COLR version 0:*

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| uint16 | version | Table version number – set to 0. |
| uint16 | numBaseGlyphRecords | Number of Base Glyph Records. |
| Offset32 | baseGlyphRecordOffset | Offset to baseGlyphRecords array. |
| Offset32 | layerRecordOffset | Offset to layerRecords array. |
| uint16 | numLayerRecords | Number of Layer Records. |

NOTE: For fonts that use COLR version 0, some early implementations of the COLR table require glyph ID 1 to be .null glyph.

*COLR version 1:*

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| uint16 | version | Table version number – set to 1. |
| uint16 | numBaseGlyphRecords | Number of Base Glyph Records, may be 0 in a version 1 table. |
| Offset32 | baseGlyphRecordOffset | Offset to baseGlyphRecords array (may be NULL). |
| Offset32 | layerRecordsOffset | Offset to layerRecords array (may be NULL). |
| uint16 | numLayerRecords | Number of Layer Records, may be 0 in a version 1 table. |
| Offset32 | baseGlyphV1ListOffset | Offset to BaseGlyphV1List table. |
| Offset32 | itemVariationStoreOffset | Offset to ItemVariationStore (may be NULL). |

The BaseGlyphV1List and its subtables are only used in COLR verions 1. The ItemVariationStore is only used in variable fonts and in conjunction with a BaseGlyphV1List and its subtables. A font that uses only BaseGlyph and Layer records should use a version 0 table.

A font that includes a BaseGlyphV1List can also include BaseGlyph and Layer records for compatibility with implementations that only support COLR version 0. For implementations that support COLR version 1, if a given base glyph is supported in the BaseGlyphV1List as well as in a BaseGlyph record, the data in the BaseGlyphV1List should be used.

Color glyphs that can be implemented in COLR version 0 using BaseGlyph and Layer records can also be implemented using the version 1 BaseGlyphV1List and subtables. Thus, a font may use the version 1 structures for some base glyphs, and version 0 structures for other base glyphs. Implementations should search for a base glyph ID first in the BaseGlyphV1List, then if not found, search in the BaseGlyph records array, if present.

**Base Glyph and Layer Record**

A BaseGlyph record is used to map a base glyph to a sequence of layer records that define the corresponding color glyph. The BaseGlyph records includes a base glyph index, and index into the layerRecords array, and the number of layers.

*BaseGlyph record:*

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| uint16 | glyphID | Glyph ID of the base glyph. |
| uint16 | firstLayerIndex | Index (base 0) into the Layer Records array. |
| uint16 | numLayers | Number of color layers associated with this glyph. |

The base glyph records are sorted by glyph id. It is assumed that a binary search can be used to efficiently access the glyph IDs that have a color glyph definition.

The color glyph for a given base glyph is defined by the consecutive records in the layerRecords array for the specified number of layers, starting with the record indicated by firstLayerIndex. The first record in this sequence is the bottom layer in the z-order, and each subsequent layer is stack on top of the previous layer.

Note that the layer record sequences for two different base glyphs can overlap, with some layer records used in multiple color glyph definitions.

The layer record specifies the glyph used as the graphic element for a layer and the solid color fill.

*Layer record:*

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| uint16 | glyphID | Glyph ID of the glyph used for a given layer. |
| uint16 | paletteIndex | Index for a palette entry in the CPAL table. |

The glyphID in a Layer record must be less than the numGlyphs value in the ‘maxp’ table (see subclause 5.2.6). That is it must be a valid glyph with the outline data in the ‘glyf’, ‘CFF ‘, or ‘CFF2’ table. The advance width of the reference glyph must be the same as that of the base glyph.

The paletteIndex value must be less that the numPaletteEntries value in the ‘CPAL’ table. A paletteIndex value of 0xFFFF is a special case, indicating that the text foreground color (as determined by the application) is to be used.

**BaseGlyphV1List and LayerV1List**

*BaseGlyphV1List table:*

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| uint32 | numBaseGlyphV1Records | Number of BaseGlyphV1 records. |
| BaseGlyphV1Record | baseGlyphV1records[numBaseGlyphV1Records] |  |

*BaseGlyphV1Record:*

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| uint16 | glyphID | Glyph ID of the base glyph. |
| Offset32 | layerListOffset | Offset to LayerV1List table, from start of BaseGlyphsV1List table. |

NOTE: The glyph ID is not limited to the numGlyphs value in the ‘maxp’ table.

The records in the baseGlyphV1Records array should be sorted in increasing glyphID order.

A LayerV1List table defines the graphic composition for a color glyph as a sequence of Paint subtables.

*LayerV1List table:*

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| uint8 | numLayers | Number of color glyph composition layers. |
| Offset32 | paintOffset[numLayers] | Offsets to Paint tables, from start of the LayerV1List table. |

NOTE: Number of simple color glyph composition layers is limited to 256 to cover most typical cases and save space. For glyphs utilizing more complex designs where larger layer counts are needed, PaintComposite can be used to combine multiple COLR v1glyphs. See Paint Tables descriptions (formats 5, 6, and 7 in particular) for more information.

***Editor’s note:*** By allowing nested COLR v1 glyphs be used as Paint layers –we may be introducing additional requirements for implementations. Please see suggested highlighted text in “Paint Tables” section to address this. Should we have a note / spec paragraph to set an upper limit the number of layers allowed in a single graph?

*Bounding box*

The bounding box of the base glph specified in the BaseGlyphV1Record is used as the bounding box for the color glyph defined in the corresponding LayerV1List. Note that a ‘glyf’ entry with two points at diagonal extrema is sufficient to define the bounding box.

NOTE: Implementations can use the bounding box to allocate a drawing surface without first needing to traverse the color glyph definition.

**Formats Used Within Paint Tables**

Variation Records

Several values contained within the Paint tables or their subtable formats are variable. These use various record formas that combine a basic data type with a variation delta-set index: VarFWord, VarUFWord, VarF2Dot14, and VarFixed. These are described in the subclause 7.2.3. “Item variation stores”.

Colors and Color lines

Colors are used in solid color fills, for graphical elements, or as stops in a color line used to define a gradient. Colors are defined by reference to palette entries in the ‘CPAL’ table. While CPAL entires include an alpha component, a *ColorIndex* record is defined here that includes a separate alpha specification that supports variation in a variable font.

*ColorIndex record:*

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| uint16 | paletteIndex | Index to a CPAL palette entry. |
| VarF2Dot14 | alpha | Variable alpha value. |

NOTE: The alpha value is always set explicitly. The alpha value, and any variations of it, should be in the range [0.0, 1.0] (inclusive); values outside this range should be clipped to the range. A value of zero means no opacity (fully transparent); value of 1.0 means fully opaque (no transparency). The alpha indicated in this record is multiplied with the alpha component of the CPAL entry. The resulting alpha value can be combined with and does not supersede alpha or opacity attributes set in higher-level contexts.

***Editor’s note:*** It may be useful to convert the text of the Note to spec proper.

A paletteIndex value of 0xFFFF is a special case, indicating that the text foreground color (as determined by the application) is to be used.

*ColorStop record:*

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| VarF2Dot14 | stopOffset | Proportional distance on a color line; variable. |
| ColorIndex | color |  |

NOTE: The stopOffset value, and any variations of it, should be in the range [0.0, 1.0] (inclusive); values outside this range should be clipped to the range. A color line is defined by array of color stops.

*ColorLine table:*

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| uint8 | extend | An Extend enum value. |
| uint16 | numStops | Number of ColorStop records. |
| ColorStop | colorStops[numStops] |  |

NOTE: The colorStops array should be in increasing stopOffset order.

***Editor’s note:*** It may be useful to convert the text of the Note to spec proper. In addition, a description of color line positioning in relation to the glyph design grid is needed.

*Extend enumeration:*

|  |  |  |
| --- | --- | --- |
| **Value** | **Name** | **Description** |
| 0 | EXTEND\_PAD | Use nearest color stop. |
| 1 | EXTEND\_REPEAT | Repeat from the farthest color stop. |
| 2 | EXTEND\_REFLECT | Mirror color line from nearest end. |

NOTE: If a ColorLine table has an unrecognized extend value, implementations should use EXTEND\_PAD by default.

Affine Transformation Matrix

A 2 x 3 affine transformation matrix is used to provide transformations of the design grid. The 2 x 3 supports scale, skew, reflection, rotation, and translation transformations. The matrix elements use VarFixed records, allowing the transform definition to be variable in a variable font.

Matrix operations are of form v’ = Mv, where v and v’ are vectors for positions in the design grid. The starting position vector v is an extended 3 x 1 column matrix with the value 1 as a third matrix element: (x,y,1). The result vector v’ is a 2 x 1 column matrix (x’,y’).

*Affine2x3 record:*

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| VarFixed | xx |  |
| VarFixed | xy |  |
| VarFixed | yx |  |
| VarFixed | yy |  |
| VarFixed | dx | Translation in x direction. |
| VarFixed | dy | Translation in y direction. |

Composition Modes

Composition modes are used to specify how two graphical compositions, one layered on top of the other, are composed together. For descriptions of supported composition modes, see W3C Compositing and Blending Level 1 specification [34]. In Paint tables, a composition mode is specified using the following enumeration.

*CompositeMode enumeration:*

|  |  |  |
| --- | --- | --- |
| **Value** | **Name** | **Description** |
| 0 | COMPOSITE\_CLEAR | See [Clear](https://www.w3.org/TR/compositing-1/#porterduffcompositingoperators_clear) |
| 1 | COMPOSITE\_SRC | See [Copy](https://www.w3.org/TR/compositing-1/#porterduffcompositingoperators_src) |
| 2 | COMPOSITE\_DEST | See [Destination](https://www.w3.org/TR/compositing-1/#porterduffcompositingoperators_dst) |
| 3 | COMPOSITE\_SRC\_OVER | See [Source Over](https://www.w3.org/TR/compositing-1/#porterduffcompositingoperators_srcover) |
| 4 | COMPOSITE\_DEST\_OVER | See [Destination Over](https://www.w3.org/TR/compositing-1/#porterduffcompositingoperators_dstover) |
| 5 | COMPOSITE\_SRC\_IN | See [Source In](https://www.w3.org/TR/compositing-1/#porterduffcompositingoperators_srcin) |
| 6 | COMPOSITE\_DEST\_IN | See [Destination In](https://www.w3.org/TR/compositing-1/#porterduffcompositingoperators_dstin) |
| 7 | COMPOSITE\_SRC\_OUT | See [Source Out](https://www.w3.org/TR/compositing-1/#porterduffcompositingoperators_srcout) |
| 8 | COMPOSITE\_DEST\_OUT | See [Destination Out](https://www.w3.org/TR/compositing-1/#porterduffcompositingoperators_dstout) |
| 9 | COMPOSITE\_SRC\_ATOP | See [Source Atop](https://www.w3.org/TR/compositing-1/#porterduffcompositingoperators_srcatop) |
| 10 | COMPOSITE\_DEST\_ATOP | See [Destination Atop](https://www.w3.org/TR/compositing-1/#porterduffcompositingoperators_dstatop) |
| 11 | COMPOSITE\_XOR | See [XOR](https://www.w3.org/TR/compositing-1/#porterduffcompositingoperators_xor) |
| 12 | COMPOSITE\_SCREEN | See [screen blend mode](https://www.w3.org/TR/compositing-1/#blendingscreen) |
| 13 | COMPOSITE\_OVERLAY | See [overlay blend mode](https://www.w3.org/TR/compositing-1/#blendingoverlay) |
| 14 | COMPOSITE\_DARKEN | See [darken blend mode](https://www.w3.org/TR/compositing-1/#blendingoverlay) |
| 15 | COMPOSITE\_LIGHTEN | See [lighten blend mode](https://www.w3.org/TR/compositing-1/#blendinglighten) |
| 16 | COMPOSITE\_COLOR\_DODGE | See [color-dodge blend mode](https://www.w3.org/TR/compositing-1/#blendingcolordodge) |
| 17 | COMPOSITE\_COLOR\_BURN | See [color-burn blend mode](https://www.w3.org/TR/compositing-1/#blendingcolorburn) |
| 18 | COMPOSITE\_HARD\_LIGHT | See [hard-light blend mode](https://www.w3.org/TR/compositing-1/#blendinghardlight) |
| 19 | COMPOSITE\_SOFT\_LIGHT | See [soft-light blend mode](https://www.w3.org/TR/compositing-1/#blendingsoftlight) |
| 20 | COMPOSITE\_DIFFERENCE | See [difference blend mode](https://www.w3.org/TR/compositing-1/#blendingdifference) |
| 21 | COMPOSITE\_EXCLUSION | See [exclusion blend mode](https://www.w3.org/TR/compositing-1/#blendingexclusion) |
| 22 | COMPOSITE\_MULTIPLY | See [multiply blend mode](https://www.w3.org/TR/compositing-1/#blendingmultiply) |
| 23 | COMPOSITE\_HSL\_HUE | See [hue blend mode](https://www.w3.org/TR/compositing-1/#blendinghue) |
| 24 | COMPOSITE\_HSL\_SATURATION | See [saturation blend mode](https://www.w3.org/TR/compositing-1/#blendingsaturation) |
| 25 | COMPOSITE\_HSL\_COLOR | See [color blend mode](https://www.w3.org/TR/compositing-1/#blendingcolor) |
| 26 | COMPOSITE\_HSL\_LUMINOSITY | See [luminosity blend mode](https://www.w3.org/TR/compositing-1/#blendingluminosity) |

Paint Tables

Seven Paint table formats are defined. Formats 1, 2, and 3 define fills. Format 4 uses a glyph outline to define a geometry. Format 5 allows an entire color glyph definition from the BaseGlyphV1List to be re-used as a component in another color glyph definition. Format 6 allows a composition, defined using a separate paint table, to be transformed. Format 7 allows compositing of two compositions, each defined using separate paint table.

***Editor’s note:*** The following statements need to be evaluated for completeness and clarity.

A color glyph definition using paint tables comprises a directed graph. This graph shall be acyclic. Since Paint format 5 allows color glyph definitions be used as component in another glyph definition, implementations should track color glyph IDs they have encountered in processing, and shall terminate the processing if a graph node referring to a previously used glyph ID (a cyclic graph) is detected.

*PaintSolid table (format 1 – solid color fill):*

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| uint8 | format | Set to 1. |
| ColorIndex | color | Solid color fill. |

*PaintLinearGradient table (format 2 – linear gradient fill):*

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| uint8 | format | Set to 2. |
| Offset24 | colorLineOffset | Offset to ColorLine, from start of PaintLinearGradient table. |
| VarFWord | x0 | Start point x coordinate. |
| VarFWord | y0 | Start point y coordinate. |
| VarFWord | x1 | End point x coordinate. |
| VarFWord | y1 | End point y coordinate. |
| VarFWord | x2 | Rotation vector end point x coordinate. |
| VarFWord | y2 | Rotation vector end point y coordinate. |

The rotation vector uses the same start point as the gradient line vector.

*PaintRadialGradient table (format 3 – radial gradient fill):*

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| uint8 | format | Set to 3. |
| Offset24 | colorLineOffset | Offset to ColorLine, from start of PaintRadialGradient table. |
| VarFWord | x0 | Start circle center x coordinate. |
| VarFWord | y0 | Start circle center y coordinate. |
| VarUFWord | radius0 | Start circle radius. |
| VarFWord | x1 | End circle center x coordinate. |
| VarFWord | y1 | End circle center y coordinate. |
| VarUFWord | radius1 | End circle radius. |

*PaintClipGlyph table (format 4):*

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| uint8 | format | Set to 4. |
| Offset24 | paintOffset | Offset to a Paint table, from the start of PaintClipGlyph table. |
| uint16 | glyphID | Glyph ID for the clip outline. |

The glyphID value must be less than the numGlyphs value in the ‘maxp’ table. That is, it must be a valid glyph with the outline data in the ‘glyf’, ‘CFF ‘ or ‘CFF2’ table.

*PaintColrGlyph table (format 5 – COLR composition):*

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| uint8 | format | Set to 5. |
| uint16 | glyphID | Virtual glyph ID for a BaseGlyphV1List base glyph. |

The glyphID value must be a glyphID found in a BaseGlyphV1Record within the BaseGlyphV1List. It may be a *virtual* glyph ID, greater than or equal the numGlyphs value in the ‘maxp’ table. The composition defined by the associated layerV1List is used as a component within the current color glyph definition.

*PaintTransform table (format 6 – transformed composition):*

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| uint8 | format | Set to 6. |
| Offset24 | paintOffset | Offset to a Paint table, from the start of PaintTransform table. |
| Affine2x3 | transform | An Affine2x3 record (inline). |

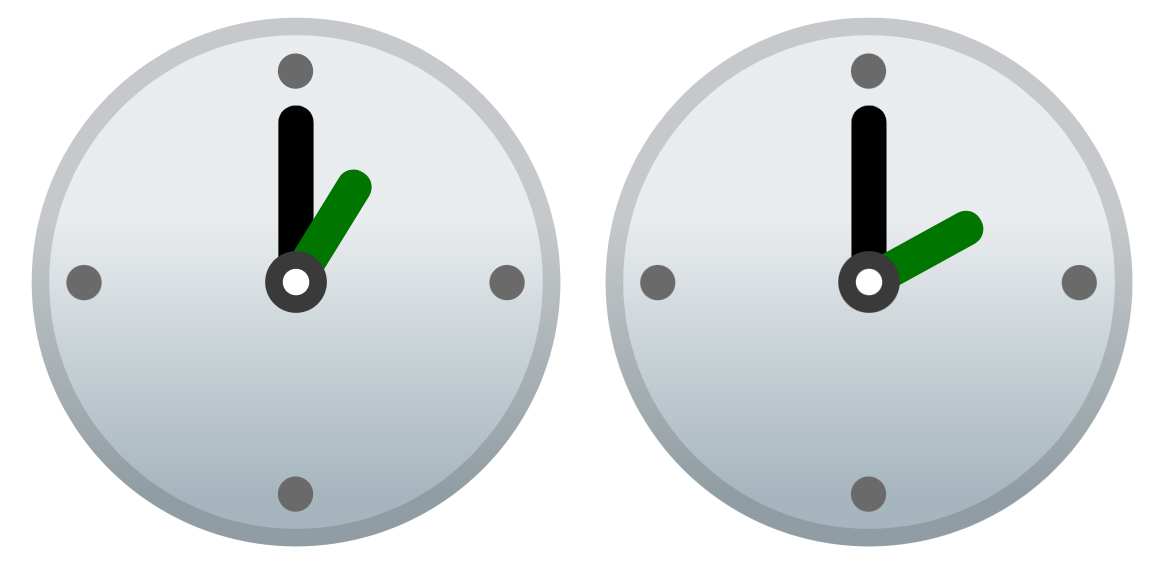
*PaintComposite table (format 7):*

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| uint8 | format | Set to 7. |
| Offset24 | sourcePaintOffset | Offset to a source Paint table, from the start of PaintComposite table. |
| uint8 | compositeMode | A CompositeMode enumeration value. |
| Offset24 | backdropPaintOffset | Offset to a backdrop Paint table, from the start of PaintComposite table. |

Format 7 is used to blend two layered compositions using different composition modes. The composition defined by the source paint table is layered on top of and blended into the destination composition defined by the backdrop paint table. The compositeMode shall be one of the value defined in the CompositeMode enumeration. If an unrecognized value is encountered, COMPOSITE\_CLEAR should be used.

*Add a new subclause 5.7.11.4 “Color glyph examples” with the following text:*

For example, consider the Noto clock emoji (hand colored for emphasis):



The entire backdrop (outline, gradient-circle, 4 dots, the minute hand) is reusable for all versions of the clock:



The hour hand is reusable as a transformed glyph.

***Editor’s note:*** Is hour hand missing in this example?

7.2.3.

*Add the following text after the first four paragraphs (prior to ItemVariationStore table description):*

**Associating Target Items to Variation Data**

The structures used in the COLR table currently are used only in that table but may be used in other tables in future versions, and so are defined here as common formats. Structures are defined to wrap the FWORD, UFWORD, F2DOT14 and Fixed basic types.

NOTE: As described below, each delta-set index is represented as two index components, an *outer* index and an *inner* index corresponding to a two-level organizational hierarchy.

*VarFWord*

The FWORD type is used to represent coordinates in the glyph design grid. The VarFWord record is used to represent a coordinate that can be variable.

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| FWORD | coordinate |  |
| uint16 | varOuterIndex |  |
| uint16 | varInnerIndex |  |

*VarUFWord*

The UFWORD type is used to represent distances in the glyph design grid. The VarUFWord record is used to represent a distance that can be variable.

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| UFWORD | distance |  |
| uint16 | varOuterIndex |  |
| uint16 | varInnerIndex |  |

*VarF2Dot14*

The F2DOT14 type is typically used to represent values that are inherently limited to a range of [-1, 1], or a range of [0, 1]. The VarF2DOT14 record is used to represent such a value that can be variable.

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| F2DOT14 | value |  |
| uint16 | varOuterIndex |  |
| uint16 | varInnerIndex |  |

NOTE: In general, variation deltas are (logically) signed 16-bit integers, and in most cases, they are applied to signed 16-bit values (FWORDs) or unsigned 16-bit values (UFWORDs). When scaled deltas are applied to F2DOT14 values, the F2DOT14 value is treated like a 16-bit integer. (In this sense, the delta and the F2DOT14 value can be viewed as an integral numerator for 1/16384ths.)

If the context in which the VarF2DOT14 is used constrains the valid range for the default value, then any variations by applying deltas are clipped to that range.

*VarFixed*

The Fixed type is intended for floating values, such as variation-space coordinates. The VarFixed record is used to represent such a value that can be variable.

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| Fixed | value |  |
| uint16 | varOuterIndex |  |
| uint16 | varInnerIndex |  |

NOTE: While in most cases deltas are applied to 16-bit types, Fixed is a 32-bit (16.16) type and requires 32-bit deltas. The DeltaSet record used in the ItemVariationData subtable format can accommodate deltas that are, logically, either 16-bit or 32-bit. See the description of the ItemVariationData subtable below, for details.  
When scaled deltas are applied to Fixed values, the Fixed value is treated like a 32-bit integer. (In this sense, the delta and the Fixed value can be viewed as an integral numerator for 1/65536ths.)

***Editor’s note:*** Should date types for varOuterIndex / varInnerIndex be defined as uint32?

**Bibliography**

*Add a new entry 32 as follows:*

[32] MDN web docs - CSS: Cascading Style Sheets  
<https://developer.mozilla.org/en-US/docs/Web/CSS>

*Add a new entry 33 as follows:*

[33] HTML Canvas specification  
<https://html.spec.whatwg.org/multipage/canvas.html#dom-context-2d-createradialgradient>

*Add a new entry 34 as follows:*

[34] W3C Candidate Recommendation: Compositing and Blending Level 1  
<https://www.w3.org/TR/compositing-1/>