COLR — Color Table

Editor's note: *The following is a draft for a revision of the COLR table spec for OT 1.9, introducing version 1 of the table. Changes shown reflect changes from version 1.8.3 of the OT spec.*

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 OpenType font files.

The COLR table defines a list of base ~~glyphs —~~glyphs, which are typically regular glyphs, ~~typically~~often associated with a 'cmap' entry. Each base glyph is associated ~~by the COLR table to a list of glyphs, each corresponding to layers that can be combined, creating a colored representation of~~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 works together with the CPAL table which holds the color palettes used by the color layers.~~

Two versions of the COLR table are defined.

Version 0 allows for a 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 elements can be glyph outlines, as in version 0. But they can also be compositions of elements, including a complete structure defined as the colored presentation for another base glyph.
- Fills are not limited to solid colors but can use different types of gradients.
- Several composition and blending modes are supported, providing options for how elements are graphically composed.

In addition, a COLR version 0 table can be used in variable fonts with glyph outlines being variable, but no other aspect of the color composition being variable. In 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.

~~Fonts using COLR and CPAL tables must implement glyph ID 1 as the .null glyph.~~ 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 ~~will not be supported for this font~~is ignored.

Processing of the COLR table is done on glyph sequences after text layout processing is completed and prior to rendering of glyphs. In the context of the COLR table, a *base glyph* is a glyph for which color presentation data is provided in this table. Typically, a base glyph is a glyph that may occur in a sequence that results from the text layout process. In some cases, a base glyph may be a virtual glyph used to define a re-usable color composition.

"Color glyph" will be used informally to refer to the graphic composition defined by the COLR data associated with a given base glyph. When a color glyph is used, it is a substitute for the base glyph: the base glyph is not presented. The same glyph ID may be used as an element in the color glyph definition, however.

The color values used in a color glyph definition are specified as entries in color palettes defined in the CPAL table. A font may define alternate palettes in its CPAL table; it is up to the application to determine which palette is used.

Graphic Compositions

The graphic compositions in a color glyph definition use a set of 2D graphic concepts and constructs:

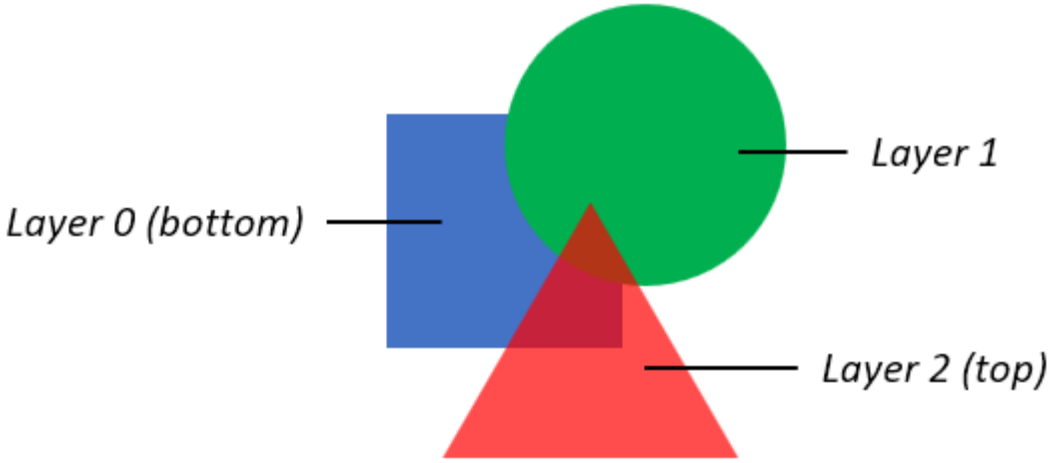
- Shapes (or *geometries*)
- Fills (or *shadings*)
- Layering—a *z-order*—of elements
- Composition modes—different ways that the content of a layer is combined with the content of layers above or below it
- Affine transformations

The simplest color glyphs use just a few of the concepts above: shapes, solid color fills, and layering. This is the set of capabilities provided by version 0 of the COLR table.

In a version 0 color glyph, a sequence of layers is defined. Each layer has a shape and a solid color fill. The shapes are obtained from glyph outlines in the 'glyf', 'CFF ' or CFF2 table. Colors are obtained from the CPAL table. The filled shapes in the layers are composed using only alpha blending.

The following figure illustrates the version 0 capabilities: three shapes are in a layered stack: a blue square in the bottom layer, an opaque green circle in the next layer, and a red triangle with some transparency in the top layer.

Editor's note: *The following figure is added. (Figures cannot be formatted to reflect changes.)*



These capabilities are sufficient to define a color glyph such as the following:

Editor's note: *The following figure is added. (Figures cannot be formatted to reflect changes.)*

The basic concepts also apply to color glyphs defined using the version 1 formats. As for version 0, all shapes are defined using glyph outlines, and all colors are obtained from the CPAL table. A sequence of layers is defined, and all shapes are arranged in layers, though there are some additional ways to incorporate layers.

Also, the version 1 concept of filling a shape is similar to that for version 0, but the fills have many more possibilities. Gradients can be used as well as solid colors. But content that fills a shape can also include *more complex compositions*. A different way to describe the relationship between a glyph outline and the way it is filled is that a fill is a graphic composition, and the glyph outline defines a bounds, or *clip region*, for the fill. This is still somewhat simplified.

More precisely, a version 1 color glyph definition is directed acyclic graph that specifies a set of nested 2D graphics operations. Glyph outlines define clip regions that apply to the nested operations that "fill" the outline. Affine transforms can be set at nodes within the graph, applying to the nested operations defined by the sub-graph. Also, composition modes can be specified at nodes within the graph determining how the composition produced by the nested operations of the sub-graph is blended into the destination surface.

All of the additional capabilities will be explained in greater detail, with examples, starting with gradients.

Gradients

<forthcoming>

Header

~~The table starts with a fixed portion describing the overall setup for the color font records. All offsets, unless otherwise noted, will be from the beginning of the table~~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 ~~(starts at 0)~~—set to 0.
uint16	numBaseGlyphRecords	Number of Base Glyph Records.
Offset32	baseGlyphRecordsOffset	Offset ~~(from beginning of COLR table) to Base Glyph records~~to baseGlyphRecords array.
Offset32	layerRecordsOffset	Offset ~~(from beginning of COLR table) to Layer Records~~to layerRecords array.
uint16	numLayerRecords	Number of Layer Records.

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

COLR version 1:

Type	Field name	Description
uint16	version	Table version number—set to 1.
uint16	numBaseGlyphRecords	May be 0 in a version 1 table.
Offset32	baseGlyphRecordsOffset	Offset to baseGlyphRecords array (may be NULL).
Offset32	layerRecordsOffset	Offset to layerRecords array (may be NULL).
uint16	numLayerRecords	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 version 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 applications that only support COLR version 0. For applications 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 that uses a BaseGlyphV1List does not need to use the version 0 BaseGlyph and Layer records. However, a font may use the version 1 structures for some base glyphs and the version 0 structures for other base glyphs. Applications 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 ~~Record~~and Layer Records

~~The header of the COLR table points to the base glyph record. This record is used to match the base glyph to the layered glyphs. Each base glyph record contains a base glyph index. This is usually the glyph index that is referenced in the 'cmap' table. The number of layers is used to indicate how many color layers will be used for this base glyph. Each record then has an index to a glyph layer record. There will be numLayers of layer records for each base glyph. The firstLayerIndex refers to the lowest z-order, or bottom, glyph id for the colored glyph. The next layer record will represent the next highest glyph in the z-order, and this continues bottom-up until it reaches the numLayers glyph at the top of the z-order. The index is relative to the start of the Layer Records. The index does not have to be unique for each base glyph ID. If two base glyphs need to share the color glyphs and palette indices, this is acceptable. Likewise a Base Glyph Record could point partway into a z-order of another base 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 color values. Any use scenario that attempts to map glyphs to character codes must be aware of the mapping done by the COLR table.~~

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

BaseGlyph record:

Type	Name	Description
uint16	~~gID~~glyphID	Glyph ID of ~~reference glyph. This glyph is for reference only and is not rendered for color~~the base glyph.
uint16	firstLayerIndex	Index ~~(from beginning of the Layer Records) to the layer record. There will be numLayers consecutive entries for this base glyph~~(base 0) into the layerRecords 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.

~~Layer Record~~

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	~~gID~~glyphID	Glyph ID of ~~layer glyph (must be in z-order from bottom to top)~~the glyph used for a given layer.
uint16	paletteIndex	~~Index value to use with a selected color palette. This value must be less than numPaletteEntries in the CPAL table. A palette entry index value of 0xFFFF is a special case indicating that the text foreground color (defined by a higher-level client) should be used and shall not be treated as actual index into CPAL ColorRecord array~~Index for a palette entry in the CPAL table.

The glyphID in a Layer record must be less than the numGlyph value in the 'maxp' table. That is, it must be a valid glyph with outline data in the 'glyf', 'CFF ' or CFF2 table. The advance width of the referenced glyph must be the same as that of the base glyph.

The paletteIndex value must be less than 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.

~~The selection of color palette to be used for a given layer record is the responsibility of a higher-level client. With CPAL version 0, the palette selection needs to be made based on the information distributed with a font. CPAL version 1 offers user-selectable color palettes based on a textual descriptions of palette entries and palette labels.~~

BaseGlyphV1List and LayerV1List

The BaseGlyphV1List table is, conceptually, similar to the baseGlyphRecords array in COLR version 0, providing records that map a base glyph to a color glyph definition. The color glyph definition is significantly different, however, defined in a LayerV1List table rather than a sequence of layer records.

BaseGlyphV1List table:

Type	Name	Description
uint32	numBaseGlyphV1Records	
BaseGlyphV1Record	baseGlyphV1Records[numBaseGlyphV1Records]	

BaseGlyphV1Record:

Type	Name	Description

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

The records in the baseGlyphV1Records array should 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	Field name	Description
uint8	numLayers	
Offset32	paintOffset[numLayers]	Offsets to Paint tables, each from the start of the LayerV1List table.

Several formats for the Paint subtable are defined, each providing a different graphic capability. A format field is the first field for each format. Specifications for each format is provided below.

Each paint table will typically have a subtable graph to define a graphic composition. The composition must define a bounded region. If a paint table in the list defines an unbounded composition, it must be ignored. See above for more details.

The sequence of offsets to paint tables corresponds to a z-order layering of the graphic compositions defined by each paint table. The first paint table defines the element at the bottom of the z-order, and each subsequent paint table defines an element that is layered on top of the previous element.

Bounding Box

The bounding box of the base glyph specified in the BaseGlyphV1Record is used at 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:* Applications can use the bounding box to allocate a drawing surface without first needing to traverse the color glyph definition.

Formats Used Within Paint Tables

Before providing specifications for the Paint table formats, various building-block elements used in paint tables will be described: variation records, colors and color lines, transforms, and composition modes.

Variation Records

Several values contained within the Paint tables or their subtable formats are variable. These use various record formats that combine a basic data type with a variation delta-set index: VarFWord, VarUFWord, VarF2Dot14, and VarFixed. These are described in the chapter, OpenType Font Variations Common Table Formats.

Colors and Color Lines

Colors are used in solid color fills for graphic 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 entries 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 for a CPAL palette entry.
VarF2Dot14	alpha	Variable alpha value.

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

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); 1.0 means opaque (no transparency). The alpha indicated in this record is multiplied with the alpha component of the CPAL entry. Note that the resulting alpha value can be combined with and does not supersede alpha or opacity attributes set in higher-level contexts.

Gradients are defined using a color line, which is a specification of color values at proportional distances from the start to the end of the line.

ColorStop record:

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

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]	

The colorStops array should be in increasing stopOffset order.

A color line defines stops at proportional distances along the line, but in a gradient specification the start and end of the line are given positions in the glyph design grid. However, the color gradation can extend beyond those limits, depending on the graphic element that is being filled. Conceptually, the color line is extended infinitely in either direction beyond the [0, 1] range. The extend field is used to indicate how the color line is extended. The same behavior is used for extension in both directions.

The extend field uses the following enumeration:

Extend enumeration:

Value	Name	Description
0	EXTEND_PAD	Use nearest color stop.
1	EXTEND_REPEAT	Repeat from farthest color stop.
2	EXTEND_REFLECT	Mirror color line from nearest end.

EXTEND_PAD: All positions on the extended color line use the color of the closest color stop. By analogy, given a sequence "ABC", it is extended to "…*AA* ABC *CC*…".

EXTEND_REPEAT: The color line is repeated by extrapolating the design grid positions in the gradient definition in either direction. In either direction, the first color in the extended color line is that of the farthest color stop. By analogy, given a sequence "ABC", it is extended to "…*ABC* ABC *ABC*…".

EXTEND_REFLECT: The color line is repeated by extrapolating the design grid positions in the gradient definition in either direction. However, the ordering of colors along the extension in either direction is reversed. For each repetition of the color line, colors are reversed again. By analogy, given a sequence "ABC", it is extended to "… *ABC CBA* ABC *CBA ABC*…".

See above for graphical illustrations of these effects.

If a ColorLine in a font has an unrecognized extend value, applications should use EXTEND_PAD by default.

Affine Transformation Matrix

A 2×3 affine transformation matrix is used to provide transformations of the design grid. The 2×3 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 the form $v' = Mv$, where v and v' are vectors for positions in the design grid. The starting position vector v is an extended 3×1 column matrix with the value 1 as a third matrix element: (x,y,1). The result vector v' is a 2×1 column matrix (x′,y′).

Affine2x3 record:

Type	Name	Description
VarFixed	xx	

Type	Name	Description
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. Supported composition modes are taken from the W3C Compositing and Blending Level 1 specification. In Paint tables, a composition mode is specified using the following enumeration.

CompositeMode enumeration:

Value	Name	Description
	Porter-Duff modes	
0	COMPOSITE_CLEAR	See Clear
1	COMPOSITE_SRC	See Copy
2	COMPOSITE_DEST	See Destination
3	COMPOSITE_SRC_OVER	See Source Over
4	COMPOSITE_DEST_OVER	See Destination Over
5	COMPOSITE_SRC_IN	See Source In
6	COMPOSITE_DEST_IN	See Destination In
7	COMPOSITE_SRC_OUT	See Source Out
8	COMPOSITE_DEST_OUT	See Destination Out
9	COMPOSITE_SRC_ATOP	See Source Atop
10	COMPOSITE_DEST_ATOP	See Destination Atop
11	COMPOSITE_XOR	See XOR
	Separable color blend modes:	
12	COMPOSITE_SCREEN	See screen blend mode
13	COMPOSITE_OVERLAY	See overlay blend mode
14	COMPOSITE_DARKEN	See darken blend mode

Value	Name	Description
15	COMPOSITE_LIGHTEN	See lighten blend mode
16	COMPOSITE_COLOR_DODGE	See color-dodge blend mode
17	COMPOSITE_COLOR_BURN	See color-burn blend mode
18	COMPOSITE_HARD_LIGHT	See hard-light blend mode
19	COMPOSITE_SOFT_LIGHT	See soft-light blend mode
20	COMPOSITE_DIFFERENCE	See difference blend mode
21	COMPOSITE_EXCLUSION	See exclusion blend mode
22	COMPOSITE_MULTIPLY	See multiply blend mode
	Non-separable color blend modes:	
23	COMPOSITE_HSL_HUE	See hue blend mode
24	COMPOSITE_HSL_SATURATION	See saturation blend mode
25	COMPOSITE_HSL_COLOR	See color blend mode
26	COMPOSITE_HSL_LUMINOSITY	See luminosity blend mode

For details on the composition modes, see the W3C specification. See above for some graphical illustrations.

Paint Tables

Seven Paint table formats (formats 1 to 7) 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 tables.

A color glyph definition using paint tables comprises a directed graph. This graph is expected to be *acyclic*. Paint format 5 creates potential for circularity by allowing the color glyph definition for a given glyph ID to reference its own glyph ID at some node in the graph. Applications should monitor the glyph ID in format 5 to see if has occurred at a higher node within the graph and, if so, ignore that sub-graph.

Paint Format 1: Solid color fill

Format 1 is used to specify a solid color fill.

PaintSolid table (format 1):

Type	Field name	Description
uint8	format	Set to 1.

Type	Field name	Description
ColorIndex	color	Solid color fill.

Paint Format 2: Linear gradient fill

Format 2 is used to specify a linear gradient fill.

PaintLinearGradient table (format 2):

Type	Field 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. See above for more information.

Paint Format 3: Radial/conic gradient fill

Format 3 is used to define a class of gradients that are a functional superset of a radial gradient: the color gradation is along a cylinder defined by two circles. In the general case, the circles can have different radii to create a conical cylinder. A radial gradient in the strict sense, with color gradation along rays from a single focal point, is formed by the starting circle having a radius of zero with center located inside the ending circle. See above for more information.

PaintRadialGradient table (format 3):

Type	Field 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.

Type	Field name	Description
VarFWord	y1	End circle center y coordinate.
VarUFWord	radius1	End circle radius.

Paint Format 4: Glyph clip region

Format 4 is used to define a clip region using a glyph outline. The outline sets a clip region that constrains the content of a separate paint subtable. Conceptually, the paint subtable defines a (potentially complex) fill for the outline.

PaintClipGlyph table (format 4):

Type	Field name	Description
uint8	format	Set to 4.
Offset24	paintOffset	Offset to a Paint table, from 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 outline data in the 'glyf', 'CFF ' or CFF2 table.

Paint Format 5: COLR composition

Format 5 is used to allow a color glyph definition from the BaseGlyphV1List to be a re-usable component in multiple color glyph definitions.

PaintColrGlyph table (format 5):

Type	Field 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 to the numGlyph value in the 'maxp' table. The composition defined by the associated LayerV1List is used as a component within the current color glyph definition.

Paint Format 6: Transformed composition

Format 6 is used to apply an affine 2×3 transform to a graphical composition defined by a separate paint table.

PaintTransformed table (format 6):

Type	Field name	Description
uint8	format	Set to 6.

Type	Field name	Description
Offset24	paintOffset	Offset to a Paint subtable, from start of PaintTransform table.
Affine2x3	transform	An Affine2x3 record (inline).

When the composition in the referenced paint table is composed into the destination (represented by the parent of this table), the source design grid origin is aligned to the destination design grid origin. The transform may translate the source such that a pre-transform position (0,0) is moved elsewhere. The post-transform origin, (0,0), is aligned to the destination origin.

Paint Format 7: Composite

Format 7 is used to blend two layered compositions using different composition modes.

PaintComposite table (format 7):

Type	Field name	Description
uint8	format	Set to 7.
Offset24	sourcePaintOffset	Offset to a source Paint table, from start of PaintComposite table.
uint8	compositeMode	A CompositeMode enumeration value.
Offset24	backdropPaintOffset	Offset to a backdrop Paint table, from start of PaintComposite table.

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 compositionMode must be one of the values defined in the CompositeMode enumeration. If an unrecognized value is encountered, COMPOSITE_CLEAR should be used.