1   The OpenType Font File

2   An OpenType font file contains data, in table format, that comprises either a TrueType
3   or a Compact Font Format (CFF) outline font. Rasterizers use combinations of data from
4   the tables contained in the font to render the TrueType or PostScript glyph outlines.
5   Some of this supporting data is used no matter which outline format is used; some of
6   the supporting data is specific to either TrueType or PostScript.

7   Filenames

8   OpenType fonts may have the extension .OTF, .TTF, .OTC or .TTC. The extensions .OTC
9   and .TTC should only be used for font collection files. For additional information on
10 filename extension conventions, see "Filenames" in Recommendations for OpenType
11 fonts.

12   Data Types

13 The following data types are used in the OpenType font file. All OpenType fonts use
14 Motorola-style byte ordering (Big Endian):

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

15 The F2DOT14 format consists of a signed, 2's complement integer and an unsigned
16 fraction. To compute the actual value, take the integer and add the fraction. Examples of
17 2.14 values are:

Decimal Value	Hex Value	Integer	Fraction
1.999939	0x7fff	1	16383/16384
1.75	0x7000	1	12288/16384
0.000061	0x0001	0	1/16384
0.0	0x0000	0	0/16384
-0.000061	0xffff	-1	16383/16384
-2.0	0x8000	-2	0/16384

18 A Tag value is a uint8 array. Each byte within the array must have a value in the range
19 0x20 to 0x7E. This corresponds to the range of values of Unicode Basic Latin characters
20 in UTF-8 encoding, which is the same as the printable ASCII characters. As a result, a Tag
21 value can be re-interpreted as a four-character sequence, which is conventionally how
22 they are referred to. Formally, however, the value is a byte array. When re-interpreted as
23 characters, the Tag value is case sensitive. It must have one to four non-space
24 characters, padded with trailing spaces (byte value 0x20). A space character cannot be
25 followed by a non-space character.

Table Version Numbers

27 Most tables have version numbers, and the version number for the entire font is
28 contained in the Table Directory. Note that there are five different version number types,
29 each with its own numbering scheme.

30 • A single uint16 field. This is used in a number of tables, usually with versions
31 starting at zero (0).
32 • Separate, uint16 major and minor version fields. This is used in a number of
33 tables, usually with versions starting at 1.0.
34 • A Fixed field for major/minor version numbers. This is used in the 'maxp', 'post'
35 and 'vhea' tables.
36 • A uint32 field with enumerated values.
37 • A uint32 field with a numeric value. This is used only in the DSIG and 'meta'
38 tables.

39   Only certain tables use a Fixed value for version, and only for reasons of backward
40   compatibility. When a Fixed number is used as a version, the upper 16 bits comprise a
41   major version number, and the lower 16 bits, a minor version. The representation of a
42   non-zero minor version, however, is not consistent with the normal treatment of Fixed
43   values, in which the lower 16 bits represent a fractional value, $N * 2 \char94 -16$. Rather, tables
44   with non-zero minor version numbers always specify the literal value of the version
45   number. For example, the version number of 'maxp' table version 0.5 is 0x00005000, and
46   that of 'vhea' table version 1.1 is 0x00011000. When Fixed is indicated as the type for a
47   version field, the possible values should be treated as an enumeration of specific values,
48   rather than as a numeric value capable of representing many potential major and minor
49   versions.

50   The Table Directory uses a uint32 field with an enumeration of defined values that
51   represent four-character tags; see the section below, "Organization of an OpenType
52   Font", for details.

53   Implementations reading tables must include code to check version numbers so that, if
54   and when the format and therefore the version number changes, older implementations
55   will handle newer versions gracefully.

56   Minor version number changes always imply format changes that are compatible
57   extensions. If an implementation understands a major version number, then it can safely
58   proceed with reading the table. If the minor version is greater than the latest version
59   recognized by the implementation, then the extension fields will be undetectable to the
60   implementation.

61   For purposes of compatibility, version numbers that are represented using a single
62   uint16 or uint32 value are treated like a minor version number, and any format changes
63   are compatible extensions.

64   Note that some field values that were undefined or reserved in an earlier revision may
65   be assigned meanings in a minor version change. Implementations should never make
66   assumptions regarding reserved or unassigned values or bits in bit fields, and can ignore
67   them if encountered. When writing font data, tools should always write zero for reserved
68   fields or bits. Validators should warn of any non-zero values for fields or bits that are not
69   defined for the given version against which data is being validated.

70   If the major version is not recognized, the implementation must not read the table as it
71   can make no assumptions regarding interpretation of the binary data. The
72   implementation should treat the table as missing.

73 Organization of an OpenType Font

74 A key characteristic of the OpenType format is the TrueType sfnt "wrapper", which
75 provides organization for a collection of tables in a general and extensible manner.

76 The OpenType font starts with the Offset Table. If the font file contains only one font,
77 the Offset Table will begin at byte 0 of the file. If the font file is an OpenType Font
78 Collection file (see below), the beginning point of the Offset Table for each font is
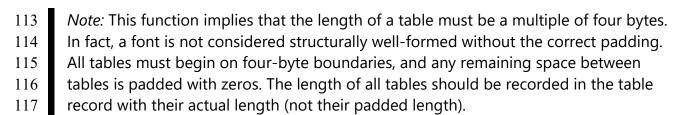79 indicated in the TTCHeader.

80 *Offset Table:*

Type	Name	Description
uint32	sfntVersion	0x00010000 or 0x4F54544F ('OTTO') — see below.
uint16	numTables	Number of tables.
uint16	searchRange	(Maximum power of 2 <= numTables) x 16.
uint16	entrySelector	Log2(maximum power of 2 <= numTables).
uint16	rangeShift	NumTables x 16-searchRange.

81 OpenType fonts that contain TrueType outlines should use the value of 0x00010000 for
82 the sfntVersion. OpenType fonts containing CFF data (version 1 or 2) should use
83 0x4F54544F ('OTTO', when re-interpreted as a Tag) for sfntVersion.

84 *Note:* The Apple specification for TrueType fonts allows for 'true' and 'typ1' for sfnt
85 version. These version tags should not be used for fonts which contain OpenType
86 tables.

87 The Offset Table is followed immediately by the Table Record entries. Entries in the
88 Table Record must be sorted in ascending order by tag. Offset values in the Table
89 Record are measured from the start of the font file.

90 *Table Record:*

Type	Name	Description
Tag	tableTag	Table identifier.
uint32	checkSum	CheckSum for this table.
Offset32	offset	Offset from beginning of TrueType font file.
uint32	length	Length of this table.

The Table Record makes it possible for a given font to contain only those tables it actually needs. As a result, there is no standard value for numTables.

Table tags are the names given to tables in the OpenType font file. For requirements of Tag values, see Data Types, above.

Some tables have an internal structure with subtables located at specified offsets, and as a result, it is possible to construct a font with data for different tables interleaved. In general, tables should be arranged contiguously without overlapping the ranges of distinct tables. In any case, however, table lengths measure a contiguous range of bytes that encompasses all of the data for a table. This applies to any subtables as well as to top-level tables.

Calculating Checksums

Table checksums are the unsigned sum of the uint32 units of a given table. In C, the following function can be used to determine a checksum:

```
uint32
CalcTableChecksum(uint32 *Table, uint32 Length)
{
uint32 Sum = 0L;
uint32 *Endptr = Table+((Length+3) & ~3) / sizeof(uint32);
while (Table < EndPtr)
    Sum += *Table++;
return Sum;
}
```

Note: This function implies that the length of a table must be a multiple of four bytes. In fact, a font is not considered structurally well-formed without the correct padding. All tables must begin on four-byte boundaries, and any remaining space between tables is padded with zeros. The length of all tables should be recorded in the table record with their actual length (not their padded length).

To calculate the checkSum for the 'head' table which itself includes the checkSumAdjustment entry for the entire font, do the following:

1. Set the checkSumAdjustment to 0.
2. Calculate the checksum for all the tables including the 'head' table and enter that value into the table directory.
3. Calculate the checksum for the entire font.
4. Subtract that value from 0xB1B0AFBA.

125   5. Store the result in checkSumAdjustment.

126   The checkSum for the 'head' table which includes the checkSumAdjustment entry for the
127   entire font is now incorrect. That is not a problem. Do not change it. An application
128   attempting to verify that the 'head' table has not changed should calculate the
129   checkSum for that table by not including the checkSumAdjustment value, and compare
130   the result with the entry in the table directory.

Font Collections

132   An OpenType Font Collection (formerly known as TrueType Collection) is a means of
133   delivering multiple OpenType font resources in a single file structure. The format for
134   font collections allows font tables that are identical between two or more fonts to be
135   shared. Font collections containing outline glyph data (TrueType, CFF, CFF2 or SVG) are
136   most useful when the fonts to be delivered together share many glyphs in common. By
137   allowing multiple fonts to share glyph sets and other common font tables, font
138   collections can result in a significant saving of file space.

139   For example, a group of Japanese fonts may each have their own designs for the kana
140   glyphs, but share identical designs for the kanji. With ordinary OpenType font files, the
141   only way to include the common kanji glyphs is to copy their glyph data into each font.
142   Since the kanji represent much more data than the kana, this results in a great deal of
143   wasteful duplication of glyph data. Font collections were defined to solve this problem.

144   *Note:* Even though the original definition of a Font Collection (as part of the TrueType
145   specification) was intended to be used with fonts containing TrueType outlines, and
146   this constraint was maintained in earlier OpenType versions, this is no longer a
147   constraint in OpenType. Font collection files may contain various types of outlines (or
148   a mix of them), regardless of whether or not fonts have layout tables present.

149   *Note:* A variable font using OpenType Font Variations mechanisms is functionally
150   equivalent to multiple non-variable fonts. Variable fonts do not need to be contained
151   within a collection file. A collection file can include one or even multiple variable fonts,
152   however, and may even combine variable and non-variable fonts.

The Font Collection File Structure

154   A font collection file consists of a single TTC Header table, one or more Offset Tables
155   with Table Directories (each corresponding to a different font resource), and a number
156   of OpenType tables. The TTC Header must be located at the beginning of the TTC file.

157   The TTC file must contain a complete Offset Table and Table Directory for each font
158   resource. A TTC file Table Directory has exactly the same format as a TTF file Table
159   Directory. The table offsets in all Table Directories within a TTC file are measured from
160   the beginning of the TTC file.

161   Each OpenType table in a TTC file is referenced through the Offset Table and Table
162   Directory of each font which uses that table. Some of the OpenType tables must appear
163   multiple times, once for each font included in the TTC; while other tables may be shared
164   by multiple fonts in the TTC.

165   As an example, consider a TTC file which combines two Japanese fonts (Font1 and
166   Font2). The fonts have different kana designs (Kana1 and Kana2) but use the same
167   design for kanji. The TTC file contains a single 'glyf' table which includes both designs of
168   kana together with the kanji; both fonts' Table Directories point to this 'glyf' table. But
169   each font's Table Directory points to a different 'cmap' table, which identifies the glyph
170   set to use. Font1's 'cmap' table points to the Kana1 region of the 'loca' and 'glyf' tables
171   for kana glyphs, and to the kanji region for the kanji. Font2's 'cmap' table points to the
172   Kana2 region of the 'loca' and 'glyf' tables for kana glyphs, and to the same kanji region
173   for the kanji.

174   The tables that should have a unique copy per font are those that are used by the
175   system in identifying the font and its character mapping, including 'cmap', 'name', and
176   OS/2. The tables that should be shared by fonts in the TTC are those that define glyph
177   and instruction data or use glyph indices to access data: 'glyf', 'loca', 'hmtx', 'hdmx',
178   LTSH, 'cvt ', 'fpgm', 'prep', EBLC, EBDT, EBSC, 'maxp', and so on. In practice, any tables
179   which have identical data for two or more fonts may be shared.

180   A tool is available from Microsoft to help build .TTC files. The process involves paying
181   close attention the issue of glyph renumbering in a font and the side effects that can
182   result, in the 'cmap' table and elsewhere. The fonts to be merged must also have
183   compatible TrueType instructions-that is, their preprograms, function definitions, and
184   control values must not conflict.

185   Collection files containing TrueType glyph outlnes should use the filename suffix .TTC.
186   Collection files containing CFF or CFF2 outlines should use the file extension .OTC.

187   ## TTC Header

188   There are two versions of the TTC Header: Version 1.0 has been used for TTC files
189   without digital signatures. Version 2.0 can be used for TTC files with *or* without digital

190 signatures — if there's no signature, then the last three fields of the version 2.0 header
191 are left null.

192 If a digital signature is used, the DSIG table for the file must be the last table in the TTC
193 file. Signatures in a TTC file are expected to be Format 1 signatures.

194 The purpose of the TTC Header table is to locate the different Offset Tables within a TTC
195 file. The TTC Header is located at the beginning of the TTC file (offset = 0). It consists of
196 an identification tag, a version number, a count of the number of OpenType fonts in the
197 file, and an array of offsets to each Offset Table.

198 *TTC Header Version 1.0:*

Type	Name	Description
TAG	ttcTag	Font Collection ID string: 'ttcf' (used for fonts with CFF or CFF2 outlines as well as TrueType outlines)
uint16	majorVersion	Major version of the TTC Header, = 1.
uint16	minorVersion	Minor version of the TTC Header, = 0.
uint32	numFonts	Number of fonts in TTC
Offset32	offsetTable[numFonts]	Array of offsets to the OffsetTable for each font from the beginning of the file

199 *TTC Header Version 2.0:*

Type	Name	Description
TAG	ttcTag	Font Collection ID string: 'ttcf'
uint16	majorVersion	Major version of the TTC Header, = 2.
uint16	minorVersion	Minor version of the TTC Header, = 0.
uint32	numFonts	Number of fonts in TTC
Offset32	offsetTable[numFonts]	Array of offsets to the OffsetTable for each font from the beginning of the file
uint32	dsigTag	Tag indicating that a DSIG table exists, 0x44534947 ('DSIG') (null if no signature)
uint32	dsigLength	The length (in bytes) of the DSIG table (null if no signature)
uint32	dsigOffset	The offset (in bytes) of the DSIG table from the beginning of the TTC file (null if no signature)

200 # Font Tables

201 The TrueType rasterizer has a much easier time traversing tables if they are padded so
202 that each table begins on a 4-byte boundary. Also, the algorithm for calculating table
203 checksums assumes that tables are 32-bit aligned. For this reason, all tables must be 32-
204 bit aligned and padded with zeroes.

205 Required Tables

206 Whether TrueType or CFF outlines are used in an OpenType font, the following tables
207 are required for the font to function correctly:

Tag	Name
'cmap'	Character to glyph mapping
'head'	Font header
'hhea'	Horizontal header
'hmtx'	Horizontal metrics
'maxp'	Maximum profile
'name'	Naming table
OS/2	OS/2 and Windows specific metrics
'post'	PostScript information

208 Tables Related to TrueType Outlines

209 For OpenType fonts based on TrueType outlines, the following tables are used:

Tag	Name
'cvt '	Control Value Table (optional table)
'fpgm'	Font program (optional table)
'glyf'	Glyph data
'loca'	Index to location
'prep'	CVT Program (optional table)
'gasp'	Grid-fitting/Scan-conversion (optional table)

210 Tables Related to CFF Outlines

211 For OpenType fonts based on CFF outlines, the following tables are used:

Tag	Name
'CFF '	Compact Font Format 1.0
CFF2	Compact Font Format 2.0
VORG	Vertical Origin (optional table)

212 It is strongly recommended that CFF OpenType fonts that are used for vertical writing
213 include a Vertical Origin (VORG) table.

214 Multiple Master support in OpenType has been discontinued as of version 1.3 of the
215 specification. The MMSD and MMFX tables that were defined in versions prior to version
216 1.3 are no longer supported.

Table Related to SVG Outlines
217

Tag	Name
'SVG '	The SVG (Scalable Vector Graphics) table

Tables Related to Bitmap Glyphs
218

Tag	Name
EBDT	Embedded bitmap data
EBLC	Embedded bitmap location data
EBSC	Embedded bitmap scaling data
CBDT	Color bitmap data
CBLC	Color bitmap location data
'sbix'	Standard bitmap graphics

219 OpenType fonts may also contain bitmaps of glyphs, in addition to outlines. Hand-tuned
220 bitmaps are especially useful in OpenType fonts for representing complex glyphs at very
221 small sizes. If a bitmap for a particular size is provided in a font, it will be used by the
222 system instead of the outline when rendering the glyph.

Advanced Typographic Tables
223

224 Several optional tables support advanced typographic functions:

Tag	Name
BASE	Baseline data
GDEF	Glyph definition data
GPOS	Glyph positioning data
GSUB	Glyph substitution data
JSTF	Justification data
MATH	Math layout data

225 For information on common table formats, please see OpenType Layout Common Table
226 Formats .

Tables used for OpenType Font Variations

Tag	Name
'avar'	Axis variations
'cvar'	CVT variations (TrueType outlines only)
'fvar'	Font variations
'gvar'	Glyph variations (TrueType outlines only)
HVAR	Horizontal metrics variations
MVAR	Metrics variations
STAT	Style attributes (required for variable fonts, optional for non-variable fonts)
VVAR	Vertical metrics variations

228 For an overview of OpenType Font Variations and the specification of the interpolation
229 algorithm used for variations, see OpenType Font Variations Overview. For details
230 regarding which tables are required or optional in variable fonts, see Variation Data
231 Tables and Miscellaneous Requirements in the Overview chapter.

232 For information on common table formats used for variations, see OpenType Font
233 Variations Common Table Formats.

234 Note that some variation-related formats may be used in tables other than the
235 variations-specific tables listed above. In particular, the GDEF or BASE tables in a variable
236 font can include variation data using common table formats. A CFF2 table in a variable
237 font can also include variation data, though using formats that are specific to the CFF2
238 table.

Tables Related to Color Fonts

Tag	Name
COLR	Color table
CPAL	Color palette table
CBDT	Color bitmap data
CBLC	Color bitmap location data
'sbix'	Standard bitmap graphics
'SVG '	The SVG (Scalable Vector Graphics) table

240 Note that several of these tables were also listed in other sections for tables related to
241 SVG outlines, and for tables related to bitmap glyphs.

242 Other OpenType Tables

Tag	Name
DSIG	Digital signature
'hdmx'	Horizontal device metrics
'kern'	Kerning
LTSH	Linear threshold data
MERG	Merge
'meta'	Metadata
STAT	Style attributes
PCLT	PCL 5 data
VDMX	Vertical device metrics
'vhea'	Vertical Metrics header
'vmtx'	Vertical Metrics

243 Note that the STAT table is required in variable fonts. Also, the 'hdmx' and VDMX tables
244 are not used in variable fonts.