Recommendations for Windows Fonts

There are many ways to construct a TrueType font file. This chapter outlines Microsoft recommendations for constructing a font file that will operate on Windows, Macintosh, and OS/2 systems, as well as with applications using Microsoft's TrueType Font Adaptation Kit. It should be noted that all TrueType fonts use Motorola-style byte ordering (Big Endian).

Filenames

The preferred suffix for TrueType font files under Windows is *.TTF.

Table Requirements & Recommendations

Table Alignment and Length

As suggested in the introduction to Chapter 2, all tables should be aligned to begin at offsets which are multiples of four bytes. While this is not required by the TrueType rasterizer, it does prevent ambiguous checksum calculations and greatly speeds table access on some processors.

All tables should be recorded in the table directory with their actual length. To ensure that checksums are calculated correctly, it is suggested that tables begin on LONG word boundries, as mentioned in Chapter 2. Any extra space after a table (and before the next LONG word boundry) should be padded with zeros.

'cmap' Table

When building a Unicode font for Windows, the platform ID should be 3 and the encoding ID should be 1 (this subtable must use cmap format 4). When building a symbol font for Windows, the platform ID should be 3 and the encoding ID should be 0.

Remember that, despite references to "first" and "second" subtables, the subtables must be stored in sorted order by platform and encoding ID.

Macintosh 'cmap' Table

When building a font containing Roman characters that will be used on the Macintosh, an additional subtable is required, specifying platform ID of 1 and encoding ID of 0 (this subtable must use cmap format 0).

In order for the Macintosh 'cmap' table to be useful, the glyphs required for the Macintosh must have glyph indices less than 256 (since the 'cmap' subtable format 0 uses BYTE indices and therefore cannot index any glyph above 255).

The Apple 'cmap' subtable should be constructed according to the guidelines in the "Character Sets" chapter. Note that the "apple logo" and "propeller" (⌘) should be mapped to the nonexistent glyph.

'cvt' Table

Should be defined only if required by font instructions.

'fpgm' Table

Should be defined only if required by font instructions.

'glyf' Table

Must contain all data required to construct the complete UGL character set as specified by the 'cmap' table.

In order for the Macintosh 'cmap' table to be useful, the glyphs required for the Macintosh must have glyph indices less than 256 (since the 'cmap' subtable format 0 uses BYTE indices and therefore cannot index any glyph above 255). This, of course, means that all the glyphs needed to map to the Macintosh character set (as per Chapter 4) must be placed within the first 256 glyph "slots" in this table.

'hdmx' Table

This table is not necessary at all unless instructions are used to control the "phantom points," and should be omitted if bit 2 of the flags field in the 'head' table is zero. (See the 'head' table documentation in Chapter 2.) Microsoft recommends that this table be included for fonts with one or more non-linearly scaled glyphs (i.e., bit 2 or 4 of the flags field is set).

Device records should be defined for all sizes from 8 through 14 point, and even point sizes from 16 through 24 point. However, the table requires pixel-per-em sizes, which depend on the horizontal resolution of the output device. The records in 'hdmx' should cover both 96 dpi devices (CGA, EGA, VGA) and 300 dpi devices (laser and ink jet printers).

Thus, 'hdmx' should contain entries for the following pixel sizes: 11, 12, 13, 15, 16, 17, 19, 21, 24, 27, 29, 32, 33, 37, 42, 46, 50, 54, 58, 67, 75, 83, 92, 100. These values have been rounded to the nearest pixel. For instance, 12 points at 300 dpi would measure 37.5 pixels, but this is rounded down to 37 for this list.

This will add approximately 9,600 bytes to the font file. However, there will be a significant improvement in speed when a client requests advance widths covered by these device records.

If the font includes an 'LTSH' table, the hdmx values are not needed above the linearity threshold.

'head' Table

All data required.

'hhea' Table

All data required. It is suggested that monospaced fonts set numberLongMetrics to three (see hmtx).

'hmtx' Table

All data required. It is suggested that monospaced fonts have three entries in the nMetric field.

'kern' Table

Should contain a single kerning pair subtable (format 0). Windows and OS/2 will not support format 2 (two-dimensional array of kern values by class). Windows and OS/2 will not support multiple tables; only the first format 0 table found will be used. Also, Windows and OS/2 will not support coverage bits 0 through 4 (i.e. assumes horizontal data, kerning values, no cross stream, and override).

‘loca’ Table

All data required. In order for the Macintosh ‘cmap’ table to be useful, the glyphs required for the Macintosh must have glyph indices less than 256 (since the ‘cmap’ subtable format 0 uses BYTE indices and therefore cannot index any glyph with an index greater than 255). Beyond this requirement, the actual ordering of the glyphs in the font can be optimized based on expected utilization, with the most frequently used glyphs appearing at the beginning of the font file. Additionally, glyphs that are often used together should be grouped together in the file. The will help to minimize the amount of swapping required when the font is loaded into memory.

‘LTSH’ Table

Should be defined if bit 2 or 4 of flags in ‘head’ is set.

‘maxp’ Table

All data required.

‘name’ Table

When building a Unicode font for Windows, the platform ID should be 3 and the encoding ID should be 1. When building a symbol font for Windows, the platform ID should be 3 and the encoding ID should be 0.

When building a font containing Roman characters that will be used on the Macintosh, an additional name record is required, specifying platform ID of 1 and encoding ID of 0.

Each set of name records should appear for US English (language ID = 0x0409 for Microsoft records, language ID = 0 for Macintosh records); additional language strings for the Microsoft set of records (platform ID 3) may be added at the discretion of the font vendor.

Remember that, despite references to “first” and “second,” the name record must be stored in sorted order (by platform ID, encoding ID, language ID, name ID). The ‘name’ table platform/encoding IDs must match the ‘cmap’ table platform/encoding IDs, which is how Windows knows which name set to use.

Name strings

The Subfamily string in the ‘name’ table should be used for variants of weight (ultra light to extra black) and style (oblique/italic or not). So, for example, the full font name of “Helvetica Narrow Italic” should be defined as Family name “Helvetica Narrow” and Subfamily “Italic.” This is so that Windows can group the standard four weights of a font in a reasonable fashion for non-typographically aware applications which only support combinations of “bold” and “italic.”

‘OS/2’ Table

All data required.

‘post’ Table

All information required, although the VM Usage fields may be set to zero. Format 2 is required in order to support the two-byte glyph indices in the UGL character set. Glyph names for the PostScript character set must be defined as per the “PostScript Reference Manual” (Adobe Systems Incorporated, 1988); note that names for all glyphs must be supplied as it cannot be assumed that all Microsoft platforms will support the default names supplied on the Macintosh. Names for the Unicode glyphs outside the PostScript set should be assigned a four character hexidecimal string that corresponds to their Unicode index (e.g. ‘2302’ for the small house glyph). See Chapter 4 for the complete Unicode character set, including PostScript glyph names (where defined).

‘prep’ Table

Should be defined only if required by the font instructions.

‘VDMX’ Table

Should be present if hints cause the font to scale non-linearly. If not present, the font is assumed to scale linearly. Clipping may occur if values in this table are absent and font exceeds linear height.

General Recommendations

Non-Standard Fonts

Non-standard fonts such as Symbol or Wingdings™ have special requirements for Microsoft platforms. These requirements affect the ‘cmap’ and ‘name’ tables; the requirements and recommendations for all other tables remain the same.

For the Macintosh, non-standard fonts can continue to use platform ID 1 (Macintosh) and encoding ID 0 (Roman character set). The ‘cmap’ subtable should use format 0 and follow the standard PostScript character encodings.

For non-standard fonts on Microsoft platforms, however, the ‘cmap’ and ‘name’ tables must use platform ID 3 (Microsoft) and encoding ID 0 (Unicode, non-standard character set). Remember that ‘name’ table encodings should agree with the ‘cmap’ table.

The Microsoft ‘cmap’ subtable (platform 3, encoding 0) must use format 4. The character codes should start at 0xF000, which is in the Private Use Area of Unicode. Microsoft suggests deriving the format 4 (Microsoft) encodings by simply adding 0xF000 to the format 0 (Macintosh) encodings.

Under both OS/2 and Windows, only the first 224 characters of non-standard fonts will be accessible: a space and up to 223 printing characters. It does not matter where in user space these start, but 0xF020 is suggested. The usFirstCharIndex and usLastCharIndex values in the ‘OS/2’ table would be set based on the actual minimum and maximum character indices used.

Device Resolutions

OS/2 and Windows make use of a logical device resolution. The physical resolution of a device is also available, but fonts will be rendered based on the logical resolution. The table below lists some important logical resolutions in dots per inch (Horizontal x Vertical). The most important ratios (in order) are 1:1, 1.67:1 and 1.33:1.

Device	Resolution	Aspect Ratio
CGA	96 x 48	2:1
EGA	96 x 72	1.33:1
VGA	96 x 96	1:1
8514	120 x 120	1:1
Dot Matrix	120 x 72	1.67:1
Laser Printer	300 x 300 or 600 x 600	1:1

Baseline to Baseline Distances

The suggested Baseline to Baseline Distance (BTBD) is computed differently for Windows and the Macintosh, and it is based on different TrueType metrics. However, if the recommendations below are followed, the BTBD will be the same for both Windows and the Mac.

Windows

Windows Metric[1]	TrueType Metric
ascent	usWinAscent
descent	usWinDescent
internal leading	usWinAscent + usWinDescent - unitsPerEm
external leading	MAX(0, LineGap - ((usWinAscent + usWinDescent) - (Ascender - Descender)))

Suggested BTBD = *ascent* + *descent* + *external leading*
It should be clear that the "external leading" can never be less than zero. Pixels above the ascent or below the descent will be clipped from the character; this is true for all output devices.

[1] These metrics are returned as part of the logical font data structure by the GDI CreateLogFont() API.

The usWinAscent and usWinDescent are values from the 'OS/2' table. The unitsPerEm value is from the 'head' table. The LineGap, Ascender and Descender values are from the 'hhea' table.

Macintosh

Ascender and Descender are metrics defined by Apple and are *not* to be confused with the Windows ascent or descent, nor should they be confused with the true typographic ascender and descender that are found in AFM files.

Macintosh Metric[2]	TrueType Metric
ascender	Ascender
descender	Descender
leading	LineGap

Suggested BTBD = *ascender + descender + leading*
If pixels extend above the ascent or below the descent, the character will be squashed in the vertical direction so that all pixels fit within these limitations; this is true for screen display only.

Making Them Match

If you perform some simple algebra, you will see that the suggested BTBD across both Macintosh and Windows will be identical if and only if:

LineGap >= (yMax - yMin) - (Ascender - Descender)

Style Bits

For backwards compatibility with previous versions of Windows, the macStyle bits in the 'head' table will be used to determine whether or not a font is regular, bold or italic (in the absence of an 'OS/2' table). This is completely independent of the usWeightClass and PANOSE information in the 'OS/2' table, the ItalicAngle in the 'post' table, and all other related metrics. If the 'OS/2' table is present, then the fsSelection bits are used to determine this information.

[2] These metrics are returned by the Mac QuickDraw GetFontInfo() API.

Drop-out Control

Drop-out control is needed if there is a difference in bitmaps with dropout control on and off. Two cases where drop-out control is needed are when the font is rotated or when the size of the font is at or below 8 ppem. Do not use SCANCTRL unless needed. SCANCTRL or the drop-out control rasterizer should be avoided for Roman fonts above 8 points per em (ppem) when the font is not under rotation. SCANCTRL should not be used for "stretched" fonts (e.g. fonts displayed at non-square aspect ratios, like that found on an EGA).

Embedded bitmaps

Three new tables are used to embed bitmaps in TrueType fonts. They are the 'EBLC' table for embedded bitmap locators, the 'EBDT' table for embedded bitmap data, and the 'EBSC' table for embedded bitmap scaling information. TrueType embedded bitmaps are also called 'sbits'.

The behavior of sbits within a TrueType font is essentially transparent to the client. A client need not be aware whether the bitmap returned by the rasterizer comes from an sbit or from a scan-converted outline.

The metrics in 'sbit' tables overrule the outline metrics at all sizes where sbits are defined. Fonts with 'hdmx' tables should correct those tables with 'sbit' values.

'Sbit only' fonts, that is fonts with embedded bitmaps but without outline data, are permitted. Care must be taken to ensure that all required TrueType tables except 'glyf' and 'loca' are present in such a font. Obviously, such fonts will only be able to return glyphs and sizes for which sbits are defined.

TrueType Collection (TTC) Files

A TrueType Collection (TTC) is a means of delivering multiple TrueType fonts in a single file structure. TrueType Collections are most useful when the fonts to be delivered together share many glyphs in common. By allowing multiple fonts to share glyph sets, TTCs can result in a significant saving of file space.

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

TTC File Structure

A TrueType Collection file consists of a single TTC Header table, two or more Table Directories, and a number of TrueType tables.

The TTC Header must be located at the beginning of the TTC file.

The TTC file must contain a complete Table Directory for each different font design. A TTC file Table Directory has exactly the same format as a TTF file Table Directory. The table offsets in all Table Directories within a TTC file are measured from the beginning of the TTC file.

Each TrueType table in a TTC file is referenced through the Table Directories of all fonts which use that table. Some of the TrueType tables must appear multiple times, once for each font included in the TTC; while other tables should be shared by all fonts in the TTC.

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

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

Creating a TrueType Collection by combining existing TrueType font files is a non-trivial process. It involves paying close attention the issue of glyph renumbering in a font and the side effects that can result, in the ‘cmap’ table and elsewhere. The fonts to be merged must also have compatible TrueType instructions—that is, their preprograms, function definitions, and control values must not conflict.

TrueType Collection files use the filename suffix .TTC.

TTC Header Table

The purpose of the TTC Header table is to locate the different Table Directories within a TTC file.

The TTC Header is located at the beginning of the TTC file (offset = 0). It consists of an identification tag, a version number, a count of the number of TrueType fonts (Table Directories) in the file, and an array of offsets to each Table Directory.

TTC Header Table

Type	Name	Description
TAG	TTCTag	TrueType Collection ID string: ‘ttcf’
FIXED32	Version	Version of the TTC Header table (initially 0x00010000)
ULONG	DirectoryCount	Number of Table Directories in TTC
ULONG	⇨ TableDirectory [DirectoryCount]	Array of offsets to Table Directories from file begin

Note that the TTC Header is *not* a table within a TrueType font file.