INTERNATIONAL STANDARD

ISO/IEC 14496-22

First edition
2007-03-15

Information technology — Coding of audio-visual objects —

Part 22: Open Font Format

Technologies de l'information — Codage des objets audiovisuels — Partie 22: Format de police de caractères ouvert

Reference number
ISO/IEC 14496-22:2007(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Contents

Page

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

ISO/IEC 14496-22 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

ISO/IEC 14496 consists of the following parts, under the general title *Information technology — Coding of audio-visual objects*:

— *Part 1: Systems*

— *Part 2: Visual*

— *Part 3: Audio*

— *Part 4: Conformance testing*

— *Part 5: Reference software*

— *Part 6: Delivery Multimedia Integration Framework (DMIF)*

— *Part 7: Optimized reference software for coding of audio-visual objects* [Technical Report]

— *Part 8: Carriage of ISO/IEC 14496 contents over IP networks*

— *Part 9: Reference hardware description* [Technical Report]

— *Part 10: Advanced Video Coding*

— *Part 11: Scene description and application engine*

— *Part 12: ISO base media file format*

— *Part 13: Intellectual Property Management and Protection (IPMP) extensions*

— *Part 14: MP4 file format*

— *Part 15: Advanced Video Coding (AVC) file format*

— *Part 16: Animation Framework eXtension (AFX)*

— *Part 17: Streaming text format*

— *Part 18: Font compression and streaming*

— *Part 19: Synthesized texture stream*

— *Part 20: Lightweight Application Scene Representation (LASeR) and Simple Aggregation Format (SAF)*

— *Part 21: MPEG-J Graphics Framework eXtension (GFX)*

— *Part 22: Open Font Format*

— *Part 23: Symbolic Music Representation*

Introduction

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

The ISO and IEC take no position concerning the evidence, validity and scope of this patent right.

The holder of this patent right has assured ISO and IEC that he 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 IEC. Information may be obtained from the companies listed in Annex A.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights other than those identified in Annex A. ISO and 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

1 Scope

This International Standard specifies the Open Font Format (OFF) specification, the TrueType™[1] and Compact Font Format (CFF) outline formats, and the TrueType hinting language. Many references to both TrueType and PostScript®[2] exist throughout this document, as Open Font Format fonts combine the two technologies.

NOTE This International Standard is based on the OpenType®[3] version 1.4 font format specification, and is technically equivalent to that specification.

Multimedia applications require a broad range of media-related standards. In addition to the typical audio and video applications, multimedia presentations include scalable 2D graphics and text supporting all languages of the world. Faithful reproduction of scalable multimedia content requires additional components including scalable font technology. The Open Font Format is an extension of the TrueType font format, adding support for PostScript font data. OFF fonts and the operating system services which support OFF fonts provide users with a simple way to install and use fonts, whether the fonts contain TrueType outlines or CFF (PostScript) outlines.

The Open Font Format addresses the following goals:

— broader multi-platform support;

— excellent support for international character sets;

— excellent protection for font data;

— smaller file sizes to make font distribution more efficient;

— excellent support for advanced typographic control.

PostScript data included in OFF fonts may be directly rasterized or converted to the TrueType outline format for rendering, depending on which rasterizers have been installed in the host operating system. But the user model is the same: OFF fonts just work. Users will not need to be aware of the type of outline data in OFF fonts. And font creators can use whichever outline format they feel provides the best set of features for their work, without worrying about limiting a font's usability.

OFF fonts can include the OFF Layout tables, which allow font creators to design broader international and high-end typographic fonts. The OFF Layout tables contain information on glyph substitution, glyph positioning, justification and baseline positioning, enabling text-processing applications to improve text layout.

---

1) TrueType is a trademark of Apple Computer Incorporated.

2) PostScript is a registered trademark of Adobe Systems Incorporated.

3) OpenType is a registered trademark of Microsoft Corporation.

As with TrueType fonts, OFF fonts allow the handling of large glyph sets using Unicode encoding. Such encoding allows broad international support, as well as support for typographic glyph variants.

Additionally, OFF fonts may contain digital signatures, which allow operating systems and browsing applications to identify the source and integrity of font files, including embedded font files obtained in web documents, before using them. Also, font developers can encode embedding restrictions in OFF fonts which cannot be altered in a font signed by the developer.

2 Normative References

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 10646, *Information technology — Universal Multiple-Octet Coded Character Set (UCS)*

ISO/IEC 14496-18, *Information technology — Coding of audio-visual objects — Part 18: Font compression and streaming*

TrueType Instruction Set, http://www.microsoft.com/typography/otspec/ttinst.htm

Unicode 4.1.0, http://www.unicode.org/versions/Unicode4.1.0/

3 The Open Font File Format

3.1 Description

An Open font file contains data, in table format, that comprises either a TrueType or a PostScript outline font. Rasterizers use combinations of data from the tables contained in the font to render the TrueType or PostScript glyph outlines. Some of this supporting data is used no matter which outline format is used; some of the supporting data is specific to either TrueType or PostScript.

3.2 Filenames

OFF fonts may have the extension .OTF or .TTF, depending on the kind of outlines in the font and the creator's desire for compatibility on systems without native OFF support.

- In all cases, fonts with only CFF data (no TrueType outlines) always have an .OTF extension.
- Fonts containing TrueType outlines may have either .OTF or .TTF, depending on the desire for backward compatibility on older systems or with previous versions of the font. TrueType Collection fonts should have a .TTC extension whether or not the fonts have OFF layout tables present.

3.3 Data Types

The following data types are used in the OFF font file. All OFF fonts use big-endian (network byte order):

Data Type	Description
BYTE	8-bit unsigned integer.
CHAR	8-bit signed integer.
USHORT	16-bit unsigned integer.

SHORT	16-bit signed integer.
ULONG	32-bit unsigned integer.
LONG	32-bit signed integer.
Fixed	32-bit signed fixed-point number (16.16)
FUNIT	Smallest measurable distance in the em space.
FWORD	16-bit signed integer (SHORT) that describes a quantity in FUnits.
UFWORD	16-bit unsigned integer (USHORT) that describes a quantity in FUnits.
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 script, language system, feature, or baseline
GlyphID	Glyph index number, same as uint16(length = 16 bits)
Offset	Offset to a table, same as uint16 (length = 16 bits), NULL Offset = 0x0000

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

Decimal Value	Hex Value	Mantissa	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

3.4 Table Version Numbers

Most tables have version numbers, and the version number for the entire font is contained in the Table Directory. It should be noted that there are two different table version number types, each with its own numbering scheme. USHORT version numbers always start at zero (0). Fixed version numbers start at one (1.0 or 0x00010000), except where noted (EBDT, EBLC and EBSC tables).

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

When a Fixed number is used as a version, the upper 16 bits comprise a major version number, and the lower 16 bits a minor. Tables with non-zero minor version numbers always specify the literal value of the version number since the normal representation of Fixed numbers is not necessarily followed. For example, the version number of 'maxp' table version 0.5 is 0x00005000, and that of 'vhea' table version 1.1 is 0x00011000. If an implementation understands a major version number, then it can safely proceed reading the table. The minor version number indicates extensions to the format that are undetectable by implementations that do not support them.

The only exception to this is the Offset Table's sfnt version. This serves solely to identify whether the OFF font contains TrueType outlines (a value of 1.0) or CFF data (the tag 'OTTO'), as described in subclause 3.5, 'Open Font Structure.'

When a USHORT number is used to indicate version, it should be treated as though it were a minor version number; i.e., all format changes are compatible extensions.

3.5 Open Font Structure

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

The OFF font with the Offset Table. If the font file contains only one font, the Offset Table will begin at byte 0 of the file. If the font file is a TrueType collection, the beginning point of the Offset Table for each font is indicated in the TTCHeader.

Offset Table		
Type	Name	Description
Fixed	sfnt version	0x00010000 for version 1.0 or 'OTTO'.
USHORT	numTables	Number of tables.
USHORT	searchRange	(Maximum power of 2 <= numTables) x 16.
USHORT	entrySelector	Log2(maximum power of 2 <= numTables).
USHORT	rangeShift	NumTables x 16-searchRange.

OFF fonts that contain TrueType outlines should use the value of 1.0 for the sfnt version. OFF fonts containing CFF data should use the tag 'OTTO' as the sfnt version number.

3.5.1 Table Directory

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

Table Directory		
Type	Name	Description
ULONG	tag	4 -byte identifier.
ULONG	checkSum	CheckSum for this table.
ULONG	Offset	Offset from beginning of TrueType font file.
ULONG	length	Length of this table.

The Table Directory 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.

Tags are the names given to tables in the OFF font file. All tag names consist of four characters. Names with less than four letters are allowed if followed by the necessary trailing spaces. All tag names defined within a font (e.g., table names, feature tags, language tags) must be built from printing characters represented by ASCII values 32-126.

NOTE Tag names are case sensitive.

3.5.2 Calculating Checksums

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

```
ULONG
CalcTableChecksum(ULONG *Table, ULONG Length)
{
ULONG Sum = 0L;
ULONG *Endptr = Table+((Length+3) & ~3) / sizeof(ULONG);

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 proper without the correct padding. All tables must begin on four byte boundries, and any remaining space between tables is padded with zeros. The length of all tables should be recorded in the table directory 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 the hex value B1B0AFBA.
5. Store the result in checkSumAdjustment.

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

3.6 TrueType Collections

A TrueType Collection (TTC) is a means of delivering multiple OFF 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 OFF 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.

The CFF rasterizer does not currently support TTC files.

3.6.1 The TTC File Structure

The TTC File Structure

A TrueType Collection file consists of a single TTC Header table, one or more Offset Tables with Table Directories, and a number of OFF tables. The TTC Header must be located at the beginning of the TTC file.

The TTC file must contain a complete Offset Table and Table Directory for each font. 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 OFF table in a TTC file is referenced through the Offset Table and Table Directory of each font which uses that table. Some of the OFF tables must appear multiple times, once for each font included in the TTC; while other tables may be shared by multiple 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 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.

NOTE Tools are available to help build .TTC files. The process 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 pre-programs, function definitions, and control values must not conflict.

TrueType Collection files use the filename suffix .TTC.

3.6.2 TTC Header

There are two versions of the TTC Header: Version 1.0 has been used for TTC files without digital signatures. Version 2.0 can be used for TTC files with *or* without digital signatures -- if there's no signature, then the last three fields of the version 2.0 header are left null.

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

The purpose of the TTC Header table is to locate the different Offset Tables 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 OFF fonts in the file, and an array of Offsets to each Offset Table.

TTC Header Version 1.0		
Type	Name	Description
TAG	TTCTag	TrueType Collection ID string: 'ttcf'
ULONG	Version	Version of the TTC Header (1.0), 0x00010000
ULONG	numFonts	Number of fonts in TTC
ULONG	OffsetTable[numFonts]	Array of Offsets to the OffsetTable for each font from the beginning of the file

TTC Header Version 2.0		
Type	Name	Description
TAG	TTCTag	TrueType Collection ID string: 'ttcf'
ULONG	Version	Version of the TTC Header (2.0), 0x00020000
ULONG	numFonts	Number of fonts in TTC
ULONG	OffsetTable[numFonts]	Array of Offsets to the OffsetTable for each font from the beginning of the file
ULONG	ulDsigTag	Tag indicating that a DSIG table exists, 0x44534947 ('DSIG') (null if no signature)
ULONG	ulDsigLength	The length (in bytes) of the DSIG table (null if no signature)
ULONG	ulDsigOffset	The Offset (in bytes) of the DSIG table from the beginning of the TTC file (null if no signature)