International Standard

ISO/IEC 14496-22

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

Fifth edition
2026-07

Reference number
ISO/IEC 14496-22:2026(en)

COPYRIGHT PROTECTED DOCUMENT

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

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.

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 document 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 or www.iec.ch/members_experts/refdocs).

ISO and IEC draw attention to the possibility that the implementation of this document may involve the use of (a) patent(s). ISO and IEC take no position concerning the evidence, validity or applicability of any claimed patent rights in respect thereof. As of the date of publication of this document, ISO and IEC had received notice of (a) patent(s) which may be required to implement this document. However, implementers are cautioned that this may not represent the latest information, which may be obtained from the patent database available at www.iso.org/patents and https://patents.iec.ch. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

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. In the IEC, see www.iec.ch/understanding-standards.

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

This fifth edition cancels and replaces the fourth edition (ISO/IEC 14496-22:2019), which has been technically revised. It also incorporates the Amendments ISO/IEC 14496-22:2019/Amd.1:2020 and ISO/IEC 14496-22:2019/Amd.2:2023.

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

— new technology clauses were added;

— many existing clauses, subclauses, tables, figures and annexes were editorially revised.

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

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 and www.iec.ch/national-committees.

Introduction

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 a widely supported format for font data with a rich set of capabilities for digital typography. It is based on the OpenType®[1] font format, which was originally developed as an extension of the TrueType™[2] font format, using the same 'sfnt' container structure, and maintaining compatibility for fonts created following the original TrueType specification. But several additional capabilities are also supported, including the following:

— Glyph outline data can use the CFF or CFF version 2 (“CFF2”) formats, as well as the TrueType glyph format.

— Multicolour glyph presentation is supported using embedded colour bitmaps or SVG documents, or using 2D graphic compositions defined in the font using a binary format that combines outline-format glyphs with various graphics operations.

— All Unicode® characters can be supported, including supplementary-plane characters, as well as Unicode variation sequences.

— Advanced Open Font Layout tables provide the advanced typographic capabilities needed for high-quality typography as well as for international text using the wide variety of scripts supported in The Unicode Standard®.

— The mathematical typesetting table allows a font to include data required for layout of complex, math formulas.

— Open Font collection files enable multiple fonts that share common data to be housed within a single file, allowing for de-duplication of data. This is especially useful, for example, for sets of CJK (Chinese, Japanese, Korean) fonts of the same design that share most glyphs in common but that vary with locale-specific glyphs for certain characters.

— Font variations (“variable fonts”) enable glyph outlines or other font data to be variable based on one or more design-axis parameters. Whereas a collection file can contain multiple discrete, static font resources, a variable font can provide continuous variation in design along each of its axes. This can provide great flexibility for content authors and designers while also allowing the font data for an entire font family to be represented in an efficient format.

This specification is intended to be used in conjunction with other specifications described below:

— While various legacy character encoding standards are supported, it is primarily designed for use with The Unicode Standard, which provides the universal encoding for written characters and symbols, as well as specifications for how text in different scripts is to be represented.

[1] OpenType® is the registered trademark of a product supplied by Microsoft. This information is given for the convenience of users of this document and does not constitute an endorsement by ISO or IEC of this product.

[2] TrueType™ is the trademark of a product supplied by Apple Inc. This information is given for the convenience of users of this document and does not constitute an endorsement by ISO or IEC of this product.

— This specification defines Advanced Open Font Layout tables and low-level glyph substitution and positioning operations needed for high-quality typography and for correct display of Unicode text in various scripts. (See 6.1.1.) The OpenType Layout feature registry defines various features that represent specific typographic capabilities that can be supported in a font, and that are used to activate those capabilities in a given font. Many features expose optional capabilities that authors and typographers can choose to use at their discretion; for example, small cap forms, or kerning. But many other features are used to activate capabilities that are required for correct display of text; for example, required ligatures for Arabic script, or positioning of mark glyphs. Many scripts supported in Unicode have complex structural behaviors that require non-trivial operations, implemented in applications or in text layout and "shaping" libraries, to derive a correct sequence of positioned glyphs for presentation of an underlying Unicode string. The feature registry will include features that can be used in those operations. However, complete specification of shaping algorithms for different scripts is beyond the scope of this specification. Such algorithms can be proprietary, "closed-source" implementations in particular applications; or they can be defined in vendor-specific specifications or in other industry specifications.

— Support for mathematical text involving complex formulas requires a content format to describe text semantics combined with layout and presentation capabilities. This specification defines the font-specific data that would be needed for presentation. In this way, the Open Font Format can be used to implement presentation capabilities for other document format specifications, such as TEX or MathML. Certain features can be defined in the Advanced Open Font Layout feature tag registry to support math layout operations.

— Text layout involves operations within individual lines of text but also control over arrangement of lines into larger blocks within a page or similar context. This specification defines certain data that is used in block-level layout, such as default line metrics (ascent, descent, leading). Also, certain features can be defined in the Advanced Open Font Layout feature tag registry to support layout of text blocks for either horizontal or vertical layout orientation. Complete specification of block level layout is beyond the scope of this specification, however. It can be used in conjunction with other specifications, such as Unicode Standard Annex #50: Unicode Vertical Text Layout.

Some font capabilities can be subject to tailoring by applications or other higher-level protocols. For example, the TrueType instructions defined in this specification define operations used in rasterization of glyph outlines, but applications can supplement their own algorithms for final rasterization (for example, using over-sampling, or control of sub-pixel display elements) to optimize legibility of glyphs. Also, while the descriptions of features in the Advanced Open Font Layout feature tag registry can specify expected usage, applications may tailor their usage of features subject to other specifications or according to their own needs.

Information technology — Coding of audio-visual objects —

Part 22:
Open font format

1 Scope

This document specifies the Open Font Format (OFF) specification, including the TrueType and Compact Font Format (CFF) outline formats. Many references to both TrueType and PostScript exist throughout this document, as Open Font Format fonts combine the two technologies. The document defines data structures for various font tables and provides the necessary details for developers to build a font rendering and text layout/shaping engines in compliance with this document.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements 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 Coded Character Set (UCS)

ISO/IEC 15948, Information technology — Computer graphics and image processing — Portable Network Graphics: Functional specification [3]

IEC 61966-2-1/Amd 1:2003: Multimedia systems and equipment — Colour measurement and management — Part 2-1: Colour management — Default RGB colour space — sRGB.

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

The Unicode Standard, Version 15.1.0, http://www.unicode.org/versions/Unicode15.1.0/

Scalable Vector Graphics (SVG) 1.1 (Second edition), W3C Recommendation, 16 August 2011 http://www.w3.org/TR/SVG11/

3 Terms, definitions and abbreviated terms

3.1 Terms and definitions

ISO and IEC maintain terminology databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at https://www.electropedia.org/

[3] Also available as a W3C Recommendation (Reference [15]).

3.2 Abbreviated terms

ACF	average character face
ANSI	American National Standards Institute
ASCII	American Standard Code for Information Interchange
BMP	[Unicode] basic multilingual plane
BTBD	baseline to baseline distance
CFF	compact font format
CID	character identifier
CJK	Chinese Japanese Korean [characters, ideographs, fonts, etc.]
CJKV	Chinese Japanese Korean and Vietnamese
CV	control value
CVT	control value table
DLL	dynamically linked library
FDEF	function definition
GID	glyph ID
ICF	ideographic character face
IDEF	instruction definition
IETF	Internet Engineering Task Force
JIS	Japanese Industrial Standard
LTR	left to right
NLC	National Language Council of Japan
OFF	open font format
OMPL	OFF mirroring pairs list
OTF	OpenType font
PCL	printer control language
PPM, PPEM	pixels per em
RTL	right to left
TTC	TrueType collection
TTF	TrueType font
UCS	universal character set
UTF	Unicode transformation format
UVS	Unicode variation sequence
VM	virtual memory
W3C	world wide web consortium

4 The Open Font file format

4.1 Description

An Open Font file contains data, in table format, used for rendering of text. Portions of the data are used by applications to calculate the layout of text using the font; that is, the selection of glyphs and their placement within a line. Other data provide descriptions of glyphs as TrueType or Compact Font Format (CFF) outlines. Still other data can provide monochromatic or colour bitmaps, SVG documents or binary, 2D vector graphics compositions as alternate glyph descriptions. The font data also includes meta-information, such as name strings that can used to present the font as an available choice in a font-picker user interface. Each of these types of data is stored in one or more tables each designed for a particular purpose.

References to the Universal Coded Character Set and the Unicode Standard are used throughout this document; the implementations of the OFF shall conform to character encoding systems specified in ISO/IEC 10646 and the Unicode Standard and cannot meet the requirements of this document without strict adherence to these standards.

4.2 Filenames

OFF font files may have the extension .OTF, .TTF, .OTC or .TTC (the extension may be upper or lower case). The extensions .OTC and .TTC should only be used for font collection files. For additional information on filename extension conventions, see subclause 8.3.

4.3 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
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.
uint32var	Encodes a 32-bit integer (uint32) in a variable number of bytes.
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).

Data Type	Description
F4DOT12	16-bit signed fixed number with the low 12 bits of fraction (4.12).
F6DOT10	16-bit signed fixed number with the low 10 bits of fraction (6.10).
LONGDATETIME	Date and time represented in number of seconds since 12:00 midnight, January 1, 1904, UTC. 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.
TupleValues	A version of Packed Deltas supporting 32-bit values.
Offset8	8-bit offset to a table, same as uint8, NULL offset = 0x00
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
Version16Dot16	Packed 32-bit value with major and minor version numbers. (See 4.4.)

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

The F4DOT12 and F6DOT10 formats work the same way as F2DOT14 but have a larger integer and less precision in the unsigned fraction part.

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

The ***uint32var*** format is an efficient and compact encoding of uint32 values, using from one to five bytes to store a single value.

Values are decoded as follows:

If the current byte's most significant bit (0x80) is zero, return the value.

Otherwise, the most significant bits are taken to indicate the number of bytes to follow, as indicated in the following table; the values are stored with the first byte being the most significant.

Bits				**Meaning**
0x80 (1000 0000)	0x40 (0100 0000)	0x20 (0010 0000)	0x10 (0001 0000)	
zero	(any value)	(any value)	(any value)	All 7 remaining bits are the value
1	0	(any value)	(any value)	One byte follows; the low 6 bits of the first are shifted left 8 bits (multiplied by 256) and the single following byte is added.
1	1	0	(any value)	Two bytes follow; the low 5 bits of the first are shifted left by 16 bits and added, then the following byte is shifted left by 8 bits and added (bitwise "or"), and then the second following byte is added.
1	1	1	0	Three bytes follow; the low 4 bits of the first byte become the most significant bits of the new value, as above.
1	1	1	1	Four bytes follow; the low 4 bits of the first byte become the most significant bits of the new value, as above. The least significant 4 bits of the initial byte shall be zero in this case to prevent overflow.

The following pseudo-code illustrates how to read and write *uint32var*-formatted variable-byte numbers that achieve greater compression in the font encoding context.

```
def _readuint32var(data, i):
    """Read a variable-length number from data starting at index i.
       Return the number and the next index.
    """

    b0 = data[i]
    if b0 < 0x80:
        return b0, i + 1
    elif b0 < 0xC0:
```