INTERNATIONAL STANDARD

**ISO/IEC 14496-28**

First edition
2012-05-01

# Information technology — Coding of audio-visual objects —

## Part 28: Composite font representation

*Technologies de l'information — Codage des objets audiovisuels —*

*Partie 28: Représentation de la police de caractères composite*



Reference number
ISO/IEC 14496-28:2012(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.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 14496-28 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*
- *Part 8: Carriage of ISO/IEC 14496 contents over IP networks*
- *Part 9: Reference hardware description*
- *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 eXtensions (GFX)*

— *Part 22: Open Font Format*

— *Part 23: Symbolic Music Representation*

— *Part 24: Audio and systems interaction*

— *Part 25: 3D Graphics Compression Model*

— *Part 26: Audio conformance*

— *Part 27: 3D Graphics conformance*

— *Part 28: Composite font representation*

# Information technology — Coding of audio-visual objects —

# Part 28:
# Composite font representation

## 1 Scope

This part of ISO/IEC 14496 specifies the Composite Font Representation, an XML-based document format that allows combining individual component font resources into a single virtual font. Recent advances in developing the Unicode Standard and the addition of new characters that represent most of the world's languages and writing systems have resulted in a significant increase of the Unicode character repertoire, and this process is likely to continue in the future. Therefore, the ability to combine a number of individual fonts supporting different languages and Unicode ranges in a single virtual font description provides the opportunity for various users and creators of multimedia, graphics and textual content to support all of the world's languages and utilize the existing font rendering solutions that are implemented in current computing platforms and deployed in many existing devices.



## 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.

BCP 47 — Tags for Identifying Languages, <http://tools.ietf.org/html/bcp47>

Unicode Version 6.1, <http://www.unicode.org/versions/Unicode6.1.0/>

## 3 Abbreviated terms

For the purposes of this document, the following abbreviated terms apply.

| | |
|---|---|
| CFF | Compact Font Format |
| CFR | Composite Font Representation |
| CID | Character ID |
| GID | Glyph ID |
| ICU | International Components for Unicode |
| OFF | Open Font Format |
| OTF | OpenType Font |
| UCS | Universal Character Set |
| UPM | Units Per Em |
| UTF | Unicode Transformation Format |

# 4 The Composite Font Representation format

## 4.1 Description

A Composite Font Representation (CFR) resource is an XML representation describing the following essential components:

- Font name – the name(s) used to identify the CFR resource
- Basic metrics for the font – the metrics derived from all of the component fonts. Typically the largest values for a specific metric will be used
- Component list

The component list can specify two kinds of components:

- Single component definitions
- Lists of component definitions based on the order of language preference

A component definition in turn is defined by a font name (a PostScript name that uniquely identifies the component font), along with an optional transformational matrix, optional ICU (International Components for Unicode) UnicodeSet pattern strings defining the Unicode coverage for the component, and an optional character mapping (cmap) that defines how Unicode characters map to glyphs in the component font. The component font must be installed or downloaded for temporary use, and/or registered with the consumer's font system in a way that it can be found via its PostScript name. Note that an optional "LocationHint" attribute may be used as part of the component font definition to specify the location of a component font resource, whether embedded, local, or remote. The ordering of the components is significant, because for elements or attributes that are optional but not specified, their values are derived from the font resource that is referenced by the first instance of a component. Also, the first component containing a particular character mapping is used when resolving overlapping or conflicting character mappings, or when multiple languages are specified.



This format provides the basic framework to develop a composite font resource that is based purely on character ranges and language preferences. It can be used to implement a simple virtual font or a fallback font resource.



## 4.2 Filenames

CFR resources, when instantiated as a file, may have the extension .cfr.

## 4.3 Syntax and data types

Refer to the DTD that is provided in Annex A.

# 5 The Composite Font Representation structure

## 5.1 High-level overview

A CFR resource consists of a single 'PosingFont' element, and its 'Components' element specifies an arbitrary number of 'ComponentDef' or 'LanguagePreferredList' elements. A CFR resource can be instantiated as a standalone file or resource, and it can be referenced by another CFR resource. Examples of Composite Font Representation are presented in Annex B.

## 5.2 The 'PosingFont' element

Required. The 'PosingFont' element may contain zero or more 'Name' elements, an optional 'FontMetrics' element, and a single 'Components' element. It also contains two required attributes, 'name' and 'version'.

*Attribute definitions:*

```
name = "string"
```

Required. A string that uniquely identifies the CFR resource, and that follows PostScript font naming conventions.

```
version = "string"
```

Required. A string that represents the version of the CFR resource.

### 5.3 The 'Name' element

Optional. Each instance of the 'Name' element specifies an attribute that is equivalent to a specific 'name' table string of an 'sfnt' resource.

*Attribute definitions:*

```
type = "string"
```

Required. An sfnt 'name' table string identifier expressed as an integer.

```
string = "string"
```

Required. The actual content of the 'Name' element.

```
language = "string"
```

Optional. The BCP 47 tag that corresponds to the language of the 'string' attribute. The script and region can also be specified, if necessary.

### 5.4 The 'FontMetrics' element



Optional. The 'FontMetrics' element specifies various line-layout attributes that apply globally to the CFR resource.

*Attribute definitions:*



```
unitsPerEm = "string"
```

Optional. A virtual UPM (Units Per Em) of the CFR resource expressed as an integer. Its purpose is to facilitate the correct interpretation of the additional attributes. Unless otherwise specified, the values of the additional attributes are based on this UPM value.

```
xMin = "string"
```

Optional. The minimal X-axis value of the glyph bounding boxes expressed as an integer.

```
yMin = "string"
```

Optional. The minimal Y-axis value of the glyph bounding boxes expressed as an integer.

```
xMax = "string"
```

Optional. The maximum X-axis value of the glyph bounding boxes expressed as an integer.

```
yMax = "string"
```

Optional. The maximum Y-axis value of the glyph bounding boxes expressed as an integer.

```
macStyle = "string"
```

Optional. The head.macStyle value expressed as a 16-digit bit array.

```
lowestRecPPEM = "string"
```

Optional. The smallest readable size expressed in number of pixels.

```
ascender = "string"
```

Optional. The horizontal typographic ascent expressed as an integer.

```
descender = "string"
```

Optional. The horizontal typographic descent expressed as an integer.

```
lineGap = "string"
```

Optional. The horizontal typographic line gap expressed as an integer.

```
vertTypoAscender = "string"
```

Optional. The vertical typographic ascent expressed as an integer.

```
vertTypoDescender = "string"
```

Optional. The vertical typographic descent expressed as an integer.

```
vertTypoLineGap = "string"
```

Optional. The vertical typographic line gap expressed as an integer.

```
advanceWidthMax = "string"
```

Optional. The maximum horizontal advance width expressed as an integer.



```
advanceHeightMax = "string"
```

Optional. The maximum vertical advance height expressed as an integer.



```
minLeftSideBearing = "string"
```

Optional. The minimum left sidebearing value expressed as an integer.

```
minRightSideBearing = "string"
```

Optional. The minimum right sidebearing value, calculated as (advanceWdthMax - minLeftSideBearing - (xMax - xMin)), and expressed as an integer.

```
minTopSideBearing = "string"
```

Optional. The minimum top sidebearing value expressed as an integer.

```
minBottomSideBearing = "string"
```

Optional. The minimum bottom sidebearing value expressed as an integer.

```
xMaxExtent = "string"
```

Optional. The value of the formula (minLeftSideBearing + (xMax - xMin)) expressed as an integer.

```
yMaxExtent = "string"
```

Optional. The value of the formula (minTopSideBearing + (yMax - yMin)) expressed as an integer.

```
caretSlopeRise = "string"
```

Optional. Used to calculate the slope of the cursor according to the formula (caretSlopeRise ÷ caretSlopeRun). Use the value 1 for a vertical cursor, which is best for horizontal writing mode.

```
caretSlopeRun = "string"
```

Optional. Used to calculate the slope of the cursor according to the formula (caretSlopeRise ÷ caretSlopeRun). Use the value 0 for a vertical cursor, which is best for horizontal writing mode.

```
caretOffset = "string"
```

Optional. The amount by which a slanted highlight on a glyph needs to be shifted to produce the best appearance. Use the value 0 for non-slanted fonts.

```
vertCaretSlopeRise = "string"
```

Optional. Used to calculate the slope of the cursor according to the formula (vertCaretSlopeRise ÷ vertCaretSlopeRun). Use the value 0 for a horizontal cursor, which is best for vertical writing mode.

```
vertCaretSlopeRun = "string"
```

Optional. Used to calculate the slope of the cursor according to the formula (vertCaretSlopeRise ÷ vertCaretSlopeRun). Use the value 1 for a horizontal cursor, which is best for vertical writing mode.

```
vertCaretOffset = "string"
```

Optional. The amount by which a slanted highlight on a glyph needs to be shifted to produce the best appearance. Use the value 0 for non-slanted fonts.



```
italicAngle = "string"
```



Optional. The italic angle in counter-clockwise degrees from the vertical expressed as an integer. Upright text should be set to 0, and text that leans to the right should be set to a negative value.

```
underlinePosition = "string"
```

Optional. The top of the underline expressed as an integer. If it is below the baseline, a negative value should be used.

```
underlineThickness = "string"
```

Optional. The suggested value for the underline thickness expressed as an integer.

```
isFixedPitch = "string"
```

Optional. A value that specifies whether the glyphs in the CFR resource are monospaced (non-zero) or proportional (zero).

```
numGlyphs = "string"
```

Optional. The implied maximum number of glyphs in the CFR resource, which is intended to be used for the purpose of optimization.

```
weightClass = "string"
```

Optional. The visual weight of the glyphs in the CFR resource expressed as an integer.

```
widthClass = "string"
```

Optional. The relative aspect ratio of the glyphs in the CFR resource expressed as an integer.

```
familyClass = "string"
```

Optional. The IBM Font Class and the IBM Font Subclass parameter values of the CFR resource expressed as an integer.

### 5.5 The 'Components' element

Required. The 'Components' element contains one or more of either the 'ComponentDef' or 'LanguagePreferredList' elements.

### 5.6 The 'LanguagePreferredList' element

The 'LanguagePreferredList' element contains one or more 'LanguagePreferredComponentDef' elements.

### 5.7 The 'LanguagePreferredComponentDef' element

Required. The 'LanguagePreferredComponentDef' element contains a single 'Language' element and one or more 'ComponentDef' elements. If the same 'ComponentDef' element is meant to serve more than one language, multiple instances of the 'LanguagePreferredComponentDef' element may be specified.

The 'LanguagePreferredList' makes most sense if there is more than one 'LanguagePreferredComponentDef' element specified, though it is not an error if it specifies only one 'LanguagePreferredComponentDef' element.

The user's language setting determines which 'LanguagePreferredComponentDef' element to use first, and when a character that is not supported by the font specified by that 'LanguagePreferredComponentDef' element is encountered, the ordering of the 'LanguagePreferredComponentDef' elements is significant, and the first 'LanguagePreferredComponentDef' element that supports the character is used.



Furthermore, the 'LanguagePreferredComponentDef' elements in the 'LanguagePreferredList' element can be reordered according to the language preference settings of the user's system or environment. For example, a CFR resource can specify a 'LanguagePreferredList' element such as the following:



```
<LanguagePreferredList>
<LanguagePreferredComponentDef <ComponentDef name="Tokyo">, language="ja">
<LanguagePreferredComponentDef <ComponentDef name="Taipei">, language="zh-Hant">
<LanguagePreferredComponentDef <ComponentDef name="Beijing">, language="zh-Hans">
</LanguagePreferredList>
```

By default, the order of the component fonts is "Tokyo," "Taipei," then "Beijing." This may be suitable for a typical Japanese environment, but for a user in Shanghai, the preferred language would be Simplified Chinese, and the preferred order of the component fonts should be "Beijing," "Taipei," then "Tokyo," because the user's language preference is likely to be Simplified Chinese first, Traditional Chinese second, and perhaps followed by no other language preference. This reordering within the 'LanguagePreferredList' element takes place in applications or environment for which users' language settings are applicable, otherwise the order of the 'LanguagePreferredComponentDef' elements should be respected.

As described above, 'LanguagePreferredComponentDef' elements can be reordered by the consumer, but consumers cannot remove nor add them.

### 5.8 The 'Language' element

Required. The 'Language' element specifies the language that is to be applied to the 'ComponentDef' elements of the 'LanguagePreferredComponentDef' element.

*Attribute definitions:*

```
string = "string"
```