INTERNATIONAL STANDARD

ISO/IEC 14496-22

Third edition
2015-10-01

AMENDMENT 2
2017-08

Information technology — Coding of audio-visual objects —

Part 22: Open Font Format

AMENDMENT 2: Updated text layout features and implementations

Technologies de l'information — Codage des objets audiovisuels —

Partie 22: Format de police de caractères ouvert

AMENDEMENT 2: Mise à jour de l'introduction et des caractéristiques de mise en page du texte

Reference number
ISO/IEC 14496-22:2015/Amd.2:2017(E)

ISO copyright office
Ch. de Blandonnet 8 • CP 401
CH-1214 Vernier, Geneva, Switzerland
Tel. +41 22 749 01 11
Fax +41 22 749 09 47
copyright@iso.org
www.iso.org

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.

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).

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. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation on 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 the following URL: www.iso.org/iso/foreword.html.

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

Information technology — Coding of audio-visual objects —

Part 22:
Open Font Format

AMENDMENT 2: Updated text layout features and implementations

Normative references

Replace the reference to the Unicode Standard with the following text:

> Unicode 9.0, < http://www.unicode.org/versions/Unicode9.0.0/>

5.2.7.3

At the end of the "Description" field, add the following text:

> Only values from 1 to 1000 are valid.

In the "Comments" field, add the following text preceding the table:

> There may be legacy platform limitations on certain usWeightClass values. The following are commonly set values:

5.5.1

In "SVG Document Index Entry", replace the second before the last paragraph with the following:

> While SVG 1.1 requires [16] in addition to plain text encoding that conforming SVG implementations must correctly support gzip-encoded [RFC 1952] and deflate-encoded [RFC 1951] data streams, this specification requires that the SVG documents be either plain-text or gzip-encoded [RFC 1952]. The encoding of the (uncompressed) SVG document must be UTF-8. In both cases, svgDocLength encodes the length of the encoded data, not the decoded document.

5.5.4

Replace the first paragraph with the following:

> The glyph descriptions in the SVG documents are considered to be the SVG versions of the glyphs with the corresponding IDs in the CFF or glyf table. They are designed on an em specified in the head table's unitsPerEm field, as with CFF and TrueType glyphs. SVG glyph definitions will be in SVG's own y-down coordinate system, upright, with the default baseline at y=0. For example, the top of a capital letter may be at y=−800, and the bottom at y=0 (see Examples section below). It is the font engine's responsibility to translate this to the coordinate system of the rest of the OT tables and the coordinate system of the graphics environment.

5.5.5

Replace the first sentence of the second paragraph with the following:

> The font engine must apply the following user agent style sheet (or implement its functional equivalent) to SVG documents processed from the SVG table:

In "Security considerations and other glyph rendering restrictions", replace the second paragraph with the following:

> These requirements correspond to the "secure animated" and "secure static" processing modes that the SVG Integration document [17] requires font documents to be run in.

In "Security considerations and other glyph rendering restrictions", replace the third (last) paragraph with the following:

> In addition, any SVG <text> and <foreignObject> elements within a glyph description must be ignored and not rendered (see the corresponding rules in the User Agent style sheet above).

Replace the fourth and fifth paragraphs with the following:

> The font engine must support at least version 1.1 of the SVG specification (exceptions are noted in the section on glyph rendering restrictions). The version attribute in the <svg> element is present in the SVG 1.1 and 1.2 specifications, but not in SVG 2. Thus, the SVG document may not always have a version field specified. Given this approach to versioning in SVG, and given that not all implementations may support all of SVG (whether 1.1 or 2), font designers should restrict their SVG, as a practical matter, to whatever is supported by SVG-in-OT implementations they care about. Targeting the capabilities of SVG 1.1 would be the approach most likely to result in cross-implementation consistency.

5.5

Add new subclause “5.5.6 SVG glyph examples” with the following content:

> SVG glyph descriptions must be defined in SVG’s own y-down coordinate system, upright, with the default baseline at y=0. It is *always* the font engine’s responsibility to translate this into the coordinate system of the rest of the OFF font rendering environment.
>
> The SVG code in these examples is presented exactly as could be used in the SVG documents of an OFF font with SVG glyph outlines. The code is not optimized for brevity.

Example: Glyph specified directly in expected position

```
<svg id="glyph7" version="1.1" xmlns="http://www.w3.org/2000/svg">
  <defs>
    <linearGradient id="grad" x1="0%" y1="0%" x2="0%" y2="100%">
      <stop offset="0%" stop-color="darkblue" stop-opacity="1" />
      <stop offset="100%" stop-color="#00aab3" stop-opacity="1" />
    </linearGradient>
  </defs>
  <rect x="100" y="-430" width="200" height="430" fill="url(#grad)" />
  <rect x="100" y="-635" width="200" height="135" fill="darkblue" />
</svg>
```

In this example, the letter “i” is drawn directly in the +x –y quadrant of the SVG coordinate system, upright, with its baseline on the x axis, exactly where the OFF font engine expects it to be.

Example: Glyph shifted up with viewBox

```
<svg id="glyph7" version="1.1" xmlns="http://www.w3.org/2000/svg" viewBox="0 1000 1000
1000">
  <defs>
    <linearGradient id="grad" x1="0%" y1="0%" x2="0%" y2="100%">
      <stop offset="0%" stop-color="darkblue" stop-opacity="1" />
      <stop offset="100%" stop-color="#00aab3" stop-opacity="1" />
    </linearGradient>
  </defs>
  <rect x="100" y="570" width="200" height="430" fill="url(#grad)" />
  <rect x="100" y="365" width="200" height="135" fill="darkblue" />
</svg>
```

In this example, the glyph description of the letter “i” is first specified in the +x +y quadrant of the SVG coordinate system, upright, with its baseline along y=1000 in the SVG coordinate system. (This may be the natural way the SVG illustrating software positioned it.) A viewBox in the <svg> element is then used to shift it upwards by 1000 units, to end up in the position where the OFF font engine expects it to be.

The diagram is the same as in the above example.

Example: Common elements shared across glyphs in same SVG doc

```
<svg version="1.1" xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/
xlink">
  <defs>
    <linearGradient id="grad" x1="0%" y1="0%" x2="0%" y2="100%">
      <stop offset="0%" stop-color="darkblue" stop-opacity="1" />
      <stop offset="100%" stop-color="#00aab3" stop-opacity="1" />
    </linearGradient>
    <g id="i-base">
      <rect x="100" y="570" width="200" height="430" fill="url(#grad)" />
```

```
    </g>
  </defs>
  <g id="glyph2" transform="translate(0,-1000)">
    <use xlink:href="#i-base" />
  </g>
  <g id="glyph13" transform="translate(0,-1000)">
    <use xlink:href="#i-base" />
    <rect x="100" y="365" width="200" height="135" fill="darkblue" />
  </g>
  <g id="glyph14" transform="translate(0,-1000)">
    <use xlink:href="#i-base" />
    <polygon fill="darkblue" points="120,500 280,500 435,342 208,342" />
  </g>
</svg>
```

In this example, the base of the letter 'i' is shared across three glyphs, and has identifier “i-base” in the <defs> section. It represents the dotless 'i' in glyph ID 2. Glyph ID 13 adds a dot on top. Glyph ID 14 adds an acute accent on top. The diagram above shows glyph IDs 2, 13, and 14, from left to right.

Note that glyph IDs 3-12 can be defined in one or more separate SVG docs, and still allow glyph IDs 2, 13, and 14 to share the same SVG doc. For example:

SVG Document Index: numEntries=5

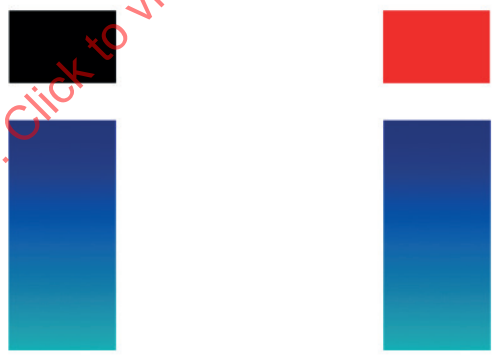
...

entries[2]: { startGlyphID = 2, endGlyphID = 2, svgDocOffset/Length point to svgDoc0 }

entries[3]: { startGlyphID = 3, endGlyphID = 12, svgDocOffset/Length point to svgDoc1 }

entries[4]: { startGlyphID = 13, endGlyphID = 14, svgDocOffset/Length point to svgDoc0 }

Example: Specifying current text color in glyphs



```
<svg id="glyph7" version="1.1" xmlns="http://www.w3.org/2000/svg" viewBox="0 1000 1000
1000">
  <defs>
    <linearGradient id="grad" x1="0%" y1="0%" x2="0%" y2="100%">
      <stop offset="0%" stop-color="darkblue" stop-opacity="1" />
      <stop offset="100%" stop-color="#00aab3" stop-opacity="1" />
    </linearGradient>
  </defs>
  <rect x="100" y="570" width="200" height="430" fill="url(#grad)" />
  <rect x="100" y="365" width="200" height="135" fill="context-fill" />
</svg>
```

Here the “darkblue” color of the dot above the “i” in the “Glyph shifted up with viewBox” example is replaced by “context-fill”. The diagram above shows the glyph when the fill color of the context element (i.e. the text color) is set to black (left) and red (right).

Example: Specifying color palette variables in glyphs

```
<svg id="glyph7" version="1.1" xmlns="http://www.w3.org/2000/svg" viewBox="0 1000 1000
1000">
  <defs>
    <linearGradient id="grad" x1="0%" y1="0%" x2="0%" y2="100%">
      <stop offset="0%" stop-color="var(--color0,darkblue)" stop-opacity="1" />
      <stop offset="100%" stop-color="var(--color1,#00aab3)" stop-opacity="1" />
    </linearGradient>
  </defs>
  <rect x="100" y="570" width="200" height="430" fill="url(#grad)" />
  <rect x="100" y="365" width="200" height="135" fill="darkblue" />
</svg>
```

This example is the duplicate of the “Glyph shifted up with viewBox” example, but with the stop colors of the linear gradient controlled by color variables --color0 and --color1, which are provided by the font engine to the SVG renderer via a user agent style sheet (or its functional equivalent).

The color palettes (CPAL) table in this font specifies two palettes, each with two color entries. Here is a description of the CPAL palettes, with alpha assumed to be 0xFF for all colors:

palette[0]: { darkblue, #00aab3 }

palette[1]: { purple, orchid }

The first item in the diagram above shows the first color palette applied to the glyph, which is done by the font engine passing the following user agent style sheet to the SVG renderer:

:root {

--color0: darkblue;

--color1: #00aab3;

}

The second item in the diagram shows the second color palette applied to the glyph, using the style sheet:

:root {

--color0: purple;

--color1: orchid;

}

Note that the dot is still dark blue, since this is hard coded in the glyph description and not controlled by a color variable.

The last item in the diagram shows the following user-selected colors applied to the glyph via the color variables:

:root {

--color0: red;

--color1: orange;

}

If --color0 and --color1 are not defined by the font engine, however, then the default values provided in the stop-colors (darkblue and #00aab3, respectively) are used. Note that these are in fact the same colors as in the first (default) CPAL color palette, which means the glyph will render as in the first item in the diagram. This way, the glyph renders with the same colors by default, whether or not the font engine supports the CPAL.

Example: Embedding a PNG in an SVG glyph


```
<svg id="glyph2" version="1.1" xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink" viewBox="0 1000 1000 1000">
  <image x="100" y="365" width="200" height="635"
  xlink:href="
iVBORw0KGgoAAAANSUhEUgAAAMgAAAJ7CAYAAACmmd5sAAAFZklEQVR42u3XsQ3D
MBAEQUpw9yppahrMPGGwiwcFMCQQW9zzWuu4FbJ2eAAQCAgGBgEBAICAQEAgIBAQC
CAQEAgIBgYBAQCAgEBAICAQEAggEBAICAYGAQEAgIBAQCAgEEAgIBAQCAgGBgEBA
ICAQEAgIBBAICAQEAgIBgYBAQCAgEBAIIBAQCAgEBAICAYGAQEAgIBAQCCAQEAgI
BAQCAgGBgEBAICAQQCAgEBAICAQEAgIBgYBAQCAgEEAgIBAQCAgEBAICAYGAQEAg
IBPAAIBgYBAQCAgEBAICAQEAgIBBAICAYGAQEAgIBAQCAgEBAICAQQCAgGBgEBA
ICAQEAgIBAQCCAQEAgIBgYBAQCAgEBAICAQEAggEBAICAYGAQEAgIBAQCAgEAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAA4DHHWtftGWDv80sE2Ds9AQgEBAL+IPBuIAoBJxYIBAQCPukgEHBigUBAIOAP
AlgQiAtiQsCCgEDAJx0sCFgQsCAgEHBigQUB5oKYELAgIBDwSQcLAhYELAgIBJxY
YEEACwItEIWAEwucWGBBwIKABQGBgBMLLAhYEMCCQFwQEwJOLHBigQUBCwICAScW
WBCwIGBBAIFAPbHcWGBBwCcdLAgIBJxYYEHAgoAFAYEA88RyY4EFAZ90sCAgEBAI
+IOAQMCJBQIBBALxD+ITAj7p4MQCgYBAwB8EBAJOLBAICATwB4EYiELAiQUCAYGA
TzoIBJxYIBAQCPiDABYE4oKYELAgIBDwSQcLAhYELAgIBJxYYEGAuSAmBCwICAR8
0sGCgAUBCwICAScWWBDAgkALRCHgxAInFlgQsCBgQUAg4MQCCwIWBLAgEBfEhIAT
C5xYYEHAgoBAwIkFFgQsCFgQQCBQTyw3FlgQ8EkHCwICAScWWBCwIGBBQCDAPLHc
WGBBwCcdLAgIBAQC/iAgEHBigUAAgUD8g/iEgE86OLFAICAQ8AcBgYATCwQCAgH8
```

```
QSAGohBwYoFAQCDgkw4CAScWCAQEAv4ggAWBuCAmBCwICAR80sGCgAUBCwICAScW
WBBgLogJAQsCAgGfdLAgYEHAgoBAwIkFFgSwINACUQg4scCJBRYELAhYEBAIOLHA
goAFASwIxAUxIeDEAicWWBCwICAQcGKBBQELAhYEEAjUE8uNBRYEfNLBgoBAwIkF
FgQsCFgQEAgwTyw3FlgQ8EkHCwICAYGAPwgIBJxYIBBAIBD/ID4h4JMOTiwQCAgE
/EFAIODEAoGAQAB/EIiBKAScWCAQEAj4pINAwIkFAgGBgD8IYEEgLogJAQsCAgGf
dLAgYEHAgoBAwIkFFgSYC2JCwIKAQMAnHSwIWBCwICAQcGKBBQEsCLRAFAJOLHBi
gQUBCwIWBAQCTiywIGBBAAsCcUFMCDixwIkFFgQsCAgEnFhgQcCCgAUBBAL1xHJj
gQUBn3SwICAQcGKBBQELAhYEBALME8uNBRYEfNLBgoBAQCDgDwICAScWCAQQCMQ/
iE8I+KSDEwsEAgIBfxAQCDixQCAgEMAfBGIgCgEnFggEBAI+6SAQcGKBQEAg4A8C
WBCIC2JCwIKAQMAnHSwIWBCwICAQcGKBBQHmgpgQsCAgEPBJBwsCFgQsCAgEnFhg
QQALAi0QhYATC5xYYEHAgoAFAYGAEwssCFgQwIJAXBATAk4scGKBBQELAgIBJxZY
ELAgYEEAgUA9sdxYYEHAJx0sCAgEnFhgQcCCgAUBgQDzxHJjgQUBn3SwICAQEAj4
g4BAwIkFAgEEAvEP4hMCPungxAKBgEDgH3wBrUwJtCBGuc0AAAAASUVORK5CYII=
  "/>
</svg>
```

In this example, the PNG is embedded using SVG's <image> element. The use case for this is bitmap lettering artwork that needs to be packaged into an OT-SVG font.

6.4.1

Replace the script tag table with the following:

Script	Script Tag
Adlam	adlm
Ahom	ahom
Anatolian Hieroglyphs	hluw
Arabic	arab
Armenian	armn
Avestan	avst
Balinese	bali
Bamum	bamu
Bassa Vah	bass
Batak	batk
Bengali	beng
Bengali v.2	bng2
Bhaiksuki	bhks
Bopomofo	bopo
Brahmi	brah
Braille	brai
Buginese	bugi
Buhid	buhd
Byzantine Music	byzm
Canadian Syllabics	cans
Carian	cari
Caucasian Albanian	aghb
Chakma	cakm
Cham	cham
Cherokee	cher
CJK Ideographic	hani
Coptic	copt
Cypriot Syllabary	cprt
Cyrillic	cyrl

Script	Script Tag
Default	DFLT
Deseret	dsrt
Devanagari	deva
Devanagari v.2	dev2
Duployan	dupl
Egyptian hieroglyphs	egyp
Elbasan	elba
Ethiopic	ethi
Georgian	geor
Glagolitic	glag
Gothic	goth
Grantha	gran
Greek	grek
Gujarati	gujr
Gujarati v.2	gjr2
Gurmukhi	guru
Gurmukhi v.2	gur2
Hangul	hang
Hangul Jamo	jamo
Hanunoo	hano
Hatran	hatr
Hebrew	hebr
Hiragana	kana
Imperial Aramaic	armi
Inscriptional Pahlavi	phli
Inscriptional Parthian	prti
Javanese	java
Kaithi	kthi
Kannada	knda
Kannada v.2	knd2
Katakana	kana
Kayah Li	kali
Kharosthi	khar
Khmer	khmr
Khojki	khoj
Khudawadi	sind
Lao	lao
Latin	latn
Lepcha	lepc
Limbu	limb
Linear A	lina
Linear B	linb
Lisu (Fraser)	lisu
Lycian	lyci
Lydian	lydi

Script	Script Tag
Mahajani	mahj
Marchen	marc
Malayalam	mlym
Malayalam v.2	mlm2
Mandaic, Mandaean	mand
Manichaean	mani
Mathematical Alphanumeric Symbols	math
Meitei Mayek (Meithei, Meetei)	mtei
Mende Kikakui	mend
Meroitic Cursive	merc
Meroitic Hieroglyphs	mero
Miao	plrd
Modi	modi
Mongolian	mong
Mro	mroo
Multani	mult
Musical Symbols	musc
Myanmar	mymr
Myanmar v.2	mym2
Nabataean	nbat
Newa	newa
New Tai Lue	talu
N'Ko	nko
Odia (formerly Oriya)	orya
Odia (formerly Oriya) v.2	ory2
Ogham	ogam
Ol Chiki	olck
Old Italic	ital
Old Hungarian	hung
Old North Arabian	narb
Old Permic	perm
Old Persian Cuneiform	xpeo
Old South Arabian	sarb
Old Turkic, Orkhon Runic	orkh
Osage	osge
Osmanya	osma
Pahawh Hmong	hmng
Palmyrene	palm
Pau Cin Hau	pauc
Phags-pa	phag
Phoenician	phnx
Psalter Pahlavi	phlp
Rejang	rjng
Runic	runr
Samaritan	samr

Script	Script Tag
Saurashtra	saur
Sharada	shrd
Shavian	shaw
Siddham	sidd
Sign Writing	sgnw
Sinhala	sinh
Sora Sompeng	sora
Sumero-Akkadian Cuneiform	xsux
Sundanese	sund
Syloti Nagri	sylo
Syriac	syrc
Tagalog	tglg
Tagbanwa	tagb
Tai Le	tale
Tai Tham (Lanna)	lana
Tai Viet	tavt
Takri	takr
Tamil	taml
Tamil v.2	tml2
Tangut	tang
Telugu	telu
Telugu v.2	tel2
Thaana	thaa
Thai	thai
Tibetan	tibt
Tifinagh	tfng
Tirhuta	tirh
Ugaritic Cuneiform	ugar
Vai	vai
Warang Citi	wara
Yi	yi

6.4.2

Add the following new language tags:

Burushaski	BSK	bsk
Medumba	BYV	byv
Fe'fe'	FMP	fmp
Krymchak	JCT	jct
Khorasani Turkic	KMZ	kmz
Mbo	MBO	mbo
Mbembe Tigon	NZA	nza
North Slavey	SCS	scs

Replace the "Slavey" language tag entry with the following:

Slavey	SLA	scs, xsl

6.4.3.2

Replace the current description of the tag 'fina' with the following text:

Tag: 'fina'

Note: This feature description was significantly revised in 2016.

Friendly name: Terminal Forms

Registered by: Microsoft/Adobe

Function: Replaces glyphs for characters that have applicable joining properties with an alternate form when occurring in a final context. This applies to characters that have one of the following Unicode Joining_Type property values:

— Right_Joining, if the characters are from a right-to-left script.

— Left_Joining, if the characters are from a left-to-right script.

— Dual_Joining.

Unicode Joining_Type property values are obtained from the Unicode Character Database (UCD) [22]. Specifically, Joining_Type property values are documented in the Unicode Character Database file for joining-script properties [20].

Example: In an Arabic-script font, the application would apply the 'fina' feature to the letter ARABIC LETTER WAW (U+0648 "و") when it follows a left-joining character, thereby replacing the default "و" glyph with its right-joining, final form.

Recommended implementation: The 'fina' feature is used to map default forms to corresponding single-joining, final forms. This will usually be implemented using a single substitution (type 1) GSUB lookup, though contextual substitution GSUB lookups (types 5, 6 or 8) may also be appropriate.

Application interface: The application is responsible for parsing character strings and identifying which of the joining-related features – initial forms ('init'), medial forms ('medi'), terminal forms ('fina'), and isolated forms ('isol') – to apply to which GIDs, based on character Joining_Type properties. Additional factors, such as the presence of control characters, may also be considered. For GIDs to which the 'fina' feature is applied and that are found in the 'fina' coverage table, the application passes a GID to the lookup tables associate with the feature and gets back a new GID.

UI suggestion: In recommended usage, this feature triggers substitutions that are required for correct display of the given script. It should be applied by default in the appropriate contexts, as determined by script-specific processing. Control of the feature should not generally be exposed to the user.

Script/language sensitivity: Can be used in any script with joining behaviour – that is, the scripts for which Joining_Type properties are given explicitly in Unicode Character Database file for joining-script properties [20].

Feature interaction: This feature may be used in combination with other substitution (GSUB) features, whose results it may override. See also 'init', 'isol', and 'medi'.

6.4.3.2

In the current description of the tag 'halt', replace the "UI suggestion" field with the following text:

UI suggestion: In general, this feature should be off by default. Different behaviour should be used, however, in applications that conform to Requirements for Japanese Text Layout (JLREQ [21]) or similar CJK text-layout specifications that expect half-width forms of characters whose default glyphs are full-width. Such implementations should turn this feature on by default, or should selectively apply this feature to particular characters that require special treatment for CJK text-layout purposes, such as brackets, punctuation and quotation marks.

6.4.3.2

In the current description of the tag 'hngl', replace the "Tag" field with the following text:

Tag: 'hngl' (DEPRECATED in 2016)

6.4.3.2

Replace the current description of the tag 'init' with the following text:

Tag: 'init'

Note: This feature description was significantly revised in 2016.

Friendly name: Initial Forms

Registered by: Microsoft/Adobe

Function: Replaces glyphs for characters that have applicable joining properties with an alternate form when occurring in an initial context. This applies to characters that have one of the following Unicode Joining_Type property values:

— Right_Joining, if the characters are from a left-to-right script.

— Left_Joining, if the characters are from a right-to-left script.

— Dual_Joining.

Unicode Joining_Type property values are obtained from the Unicode Character Database (UCD) [22]. Specifically, Joining_Type property values are documented in the Unicode Character Database file for joining-script properties [20]. Example: In an Arabic-script font, the application would apply the 'init' feature to the letter ARABIC LETTER SEEN (U+0633 “س”) when it precedes a right-joining character, thereby replacing the default “س” glyph with its left-joining, initial form.

Recommended implementation: The 'init' feature is used to map default forms to corresponding single-joining, iniital forms. This will usually be implemented using a single substitution (type 1) GSUB lookup, though contextual substitution GSUB lookups (types 5, 6 or 8) may also be appropriate.

Application interface: The application is responsible for parsing character strings and identifying which of the joining-related features – initial forms ('init'), medial forms ('medi'), terminal forms ('fina') and isolated forms ('isol') – to apply to which GIDs, based on character Joining_Type properties. Additional factors, such as the presence of control characters, may also be considered. For GIDs to which the 'init' feature is applied and that are found in the 'init' coverage table, the application passes a GID to the lookup tables associate with the feature and gets back a new GID.

UI suggestion: This feature should be active by default.

Script/language sensitivity: Can be used in any script with joining behaviour – that is, the scripts for which Joining_Type properties are given explicitly in Unicode Character Database file for joining-script properties [20].Feature interaction: This feature may be used in combination with other substitution (GSUB) features, whose results it may override. See also ‘fina’, ‘isol’, and ‘medi’.

6.4.3.2

Replace the current description of the tag 'isol' with the following text:

