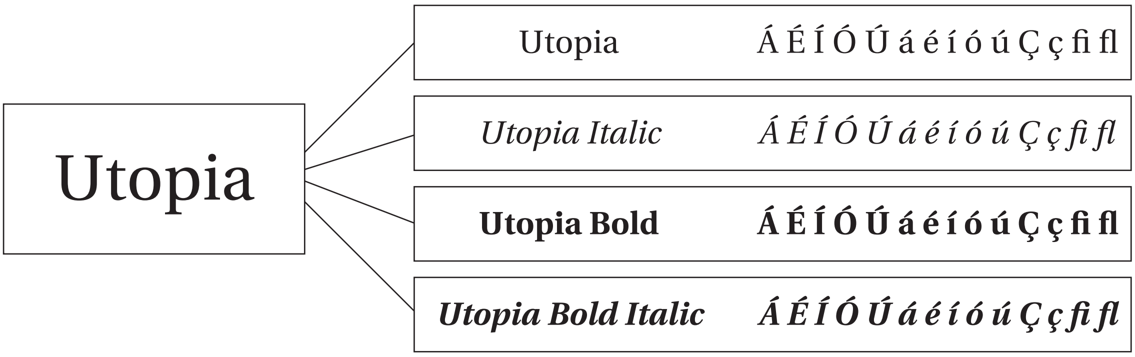
ISO/IEC JTC1/SC2/WG2 N2147
2000-01-06

Universal Multiple-Octet Coded Character Set
International Organization for Standardization
Organisation Internationale de Normalisation
Международная организация по стандартизации

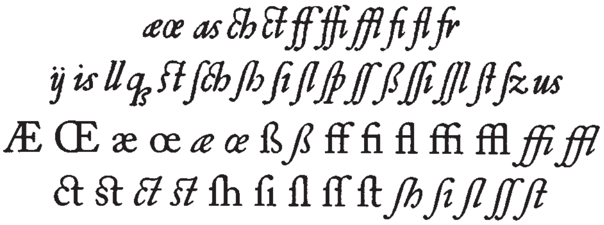
Doc Type: **Working Group Document**
Title: **Further discussion of the ZERO-WIDTH LIGATOR**
Source: **Michael Everson**
Status: **Expert Contribution**
Date: **2000-01-06**

1. Structure of font styles. A given Latin typeface typically has a minimum of four fonts associated with it: plain, *italic*, **bold**, ***bold italic***. Each of these is a self-contained unit, containing in it all the glyphs made available by the designer for the character set it handles.


Utopia
Utopia Á É Í Ó Ú á é í ó ú Ç ç fi fl
Utopia Italic Á É Í Ó Ú á é í ó ú Ç ç fi fl
Utopia Bold Á É Í Ó Ú á é í ó ú Ç ç fi fl
Utopia Bold Italic Á É Í Ó Ú á é í ó ú Ç ç fi fl

Glyphs are accessed by various input methods: they are associated with coded characters entered which often will have a many-to-one relationship with the glyphs which may be presented in individual glyph cells. Glyphs are typically precomposed units, though mechanisms may exist for dynamic presentation of character sequences which do not have, in the font, a particular precomposed glyph; *cf.* phonetic fonts where bounding-box parameters may be used to represent sequences of base characters and combining characters.

Precomposed ligatures in fonts and their respective typographic styles have existed since the time of the earliest printed typefaces. The top two lines in the sample below (taken from Bringhurst's *Elements of typographic style*) are from an italic font cut in the 1650s (350 years ago) by Cristoffel van Dijck, and the last bottom lines are ligatures from Adobe's *Caslon* roman and italic, based on William Caslon's font from 1750 (250 years ago).

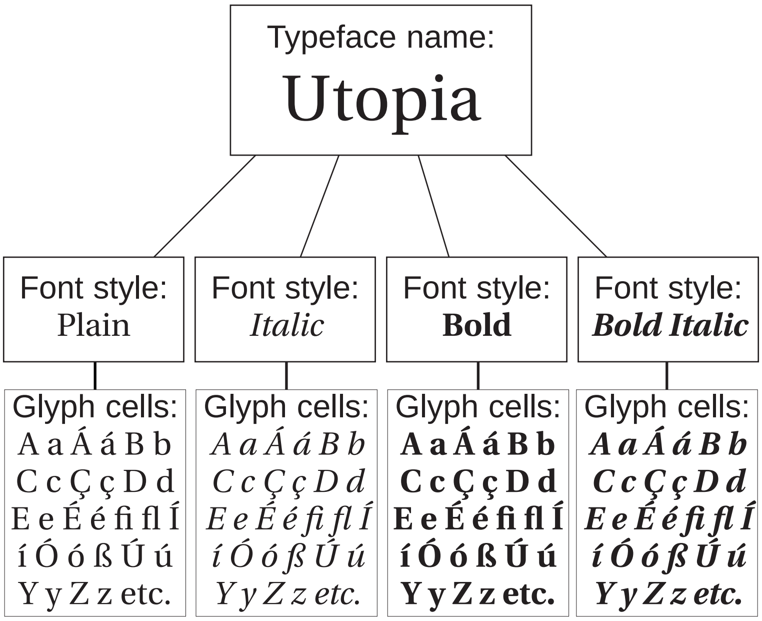


2. Input methods to access characters. Various input methods may yield the same coded character sequence: one may press the "a" key + the "ring above" deadkey, or one may press the "å" key (on a Danish keyboard), or one may press ALT–"k" + the "a" key. In all three cases the character sequence generated will be (in normalized Unicode text) LATIN SMALL LETTER A + COMBINING RING ABOVE. This inputting model is the one currently used and it works just fine. Inputting is separate from encoding but generates coded character sequences, and coded character sequences are related to glyph selection. In most cases, since *å* is a letter found in

ISO/IEC 8859-1 (a popular standard), this particular character sequence will be linked to a particular precomposed glyph cell in the font, which will have some sort of address. The specific assignment of the address to which a coded character sequence points is up to the designer and is in no way standardized or standardizable, since it is not practical to try to exhaustively specify what sequences of base characters and combining characters could be required in general, and the total requirements of a given language are only *sometimes* predictable.

3. Style selection. Style selection is much different than glyph selection. It des not involve not choosing a glyph cell *within* a particular font, but rather switching *between* a set of linked fonts (plain, *italic*, **bold**, ***bold italic***) – such selection is outside the scope of character encoding and is handled by markup of various kinds. *Within* a particular style, glyph selection should be based on the relation given by the designer to glyph cells and a set of coded characters, alone or in sequence. At present, however, *within* a particular style, only the accented-letter type is so selected; ligatures are not.

NOTE: Most people use the word *font* incorrectly. The hierarchy goes like this:


Typeface name:
Utopia
Font style: Plain
Font style: Italic
Font style: Bold
Font style: Bold Italic
Glyph cells: A a Á á B b C c Ç ç D d E e É é ﬁ ﬂ Í í Ó ó ß Ú ú Y y Z z etc.
Glyph cells: A a Á á B b C c Ç ç D d E e É é ﬁ ﬂ Í í Ó ó ß Ú ú Y y Z z etc.
Glyph cells: A a Á á B b C c Ç ç D d E e É é ﬁ ﬂ Í í Ó ó ß Ú ú Y y Z z etc.
Glyph cells: A a Á á B b C c Ç ç D d E e É é ﬁ ﬂ Í í Ó ó ß Ú ú Y y Z z etc.

4. The current model as applied to Brahmic scripts. The same set of parameters described above applies to Brahmic scripts, as in a Devanagari font. Inputting is separate from encoding but generates coded character sequences, and coded character sequence is related to glyph selection. One may press the "क" *ka* key + the "ि" *i* key, or the "ि" *i* key + the "क" *ka* key, and in either case the character sequence generated will be DEVANAGARI LETTER KA + DEVANAGARI VOWEL SIGN I, and that sequence will select a glyph cell which gives कि *ki*. Conjuncts (equivalent to ligatures) are achieved by coded character sequences, regardless of inputting: a keyboard may have a dedicated "क्ष" *kṣa* key, and pressing it will yield the coded character sequence DEVANAGARI LETTER KA + DEVANAGARI VIRAMA + DEVANAGARI LETTER SSA (क + ् + ष), and that sequence will select a single glyph cell (क्ष) in the font; or one can type explicitly pressing the "क" *ka* key, the "्" VIRAMA key, and the "ष" *ṣa* key with the same result. At another level, markup may switch to an italic or bold font; this form of switching is familiar and has been implemented for both Brahmic and European fonts.

5. The current model as applied to Arabic scripts. Arabic cursive shaping behaviour is intrinsic to the script and glyph selection with regard to cursivity is rule-governed and must be automatic. When this automatic behaviour must be overridden for some purpose, the ZERO-WIDTH JOINER and ZERO-WIDTH NON-JOINER are used, as invisible characters with cursive shaping properties, to select particular glyph cells according to the same parameters. This elegant model works very well for Arabic and other related cursive scripts like Syriac and Mongolian, though a number of script-specific characters were added for Mongolian to handle some otherwise unpredictable shaping behaviour in plain text. For the question of true Arabic ligatures, see page 19 below.

6. The current model as applied to European scripts. The current model fails in its handling of ligation in European scripts. This is because, inconsistently with the mechanism used for Brahmic scripts, certain individual glyph cells (ligature glyphs) in a font are not accessed by strings of coded characters, but rather by "some sort" of markup, while other glyph cells (accented vowels and consonants) are accessed by strings of coded characters. This is a different model than that used for shifting between font styles via markup; it forces a relationship between various undefined and ad-hoc markup levels ("turn on all ligatures", "turn on 'some' ligatures") which itself introduces a level of complexity between font designers, application programmers, and end users which is quite unsatisfactory. Quark XPress allows the Macintosh user to turn *fi* and *fl* ligatures on or off, and this works from font to font because these two ligature pairs are hard-coded in standard positions in Macintosh 8-bit fonts; but other ligatures are not recognized by XPress at all (the inquisitive reader is invited to save the present PDF file as plain text to see what I have had to do to represent all the *f*-ligatures which I have used consistently within it). The UCS also encodes a very small subset of ligatures in the FB block, but I agree that it is better for the vectoring from character sequence to glyph cell to be internal to the font, since this leads to fewer normalizable characters and addresses the difference in typefaces, some of which may have a great many ligatures and some which may have only a few.

7. Logic of the ZERO-WIDTH LIGATOR. The proposed ZERO-WIDTH LIGATOR allows font designers, programmers, and users to apply the same logic for ligatures that they apply to accented letters. Like the SOFT HYPHEN, the ZERO-WIDTH LIGATOR character may in some instances ("all ligatures on") be automatically inserted at input. This would work fine for the most common Latin ligatures, *i.e.* those in *f*-, or for a superset of those, involving *ct* and *st* ligatures and the like, available from an application according to the same criteria employed by AAT and OpenType now. Other forms of ligation, however, such as those required in the Gutenberg Bible, in modern Fraktur German, or in earlier Irish and Greek typography (for which conventions may differ from document to document even in the same typeface), can be dealt with at input by the user, or globally after inputting, by specifying ZERO-WIDTH LIGATOR insertion, or by deletion of ZERO-WIDTH LIGATOR in the case of Turkish and Azerbaijani *fi* (see page 6 below).

Glyph selection *within* a font (*Utopia* roman and *Utopia* italic are two different *fonts*) should not be handled in two different ways. Selection of the glyph *é* derives from a vector-relation between coded character sequence and glyph cell. Selection of the ligature *ffi* should make use of the same mechanism. Ligation in European scripts is *not* a style attribute. It is a glyph-selection attribute, and style is a different level of abstraction. Unicode is currently inconsistent in the way it specifies European ligatures and the way it specifies Brahmic ligatures.

Code			*Glyph*
0065 0301	e + ◌́	é	Latin
0915 093F	क + ि	कि *ki*	Devanagari
0066 ZWL 006C	f + ZWL + i	ﬁ	Latin
0915 094D 0937	क + ् + ष	क्ष *kṣa*	Devanagari
16DE ZWL 16DE	ᛞ + ZWL + ᛞ	ᛞᛞ *dd*	Runic
168F ZWL 168F	ᚏ + ZWL + ᚏ	ᚏᚏ *rr*	Ogham
xx0A ZWL xx00	𐲍 + ZWL + 𐲀	𐲍𐲀 *ga*	Old Hungarian
xx04 ZWL xx0D ZWL xx15	𐲇 + ZWL + 𐲐 + ZWL + 𐲙	𐲇𐲐𐲙 *csin*	Old Hungarian
xx00 ZWL xx14 ZWL xx02	𐲀 + ZWL + 𐲘 + ZWL + 𐲂	𐲀𐲘𐲂 *amb*	Old Hungarian
03C0 ZWL 03B1 ZWL 03C1 ZWL 03B1	π + ZWL + α + ZWL + ρ + ZWL + α	παρα *para*	Greek
0574 ZWL 0576	մ + ZWL + ն	ﬓ *mn*	Armenian

8. Example of Runic ligatures (bind-runes). A 9th-century Old English inscription from Thornhill, Yorkshire, showing a single ᛞᛞ *dd* ligature:

ᛄᛁᛚᛋᚢ'ᚦ᛬ ᚪᚱᚫᚱᛞᛖ᛬ ᚫᚠᛏᛖ ᛒᛖᚱᚻᛏᛋᚢ'ᚦᛖ᛫ ᛒᛖᛣᚢᚾ ᚩᚾᛒᛖᚱᚷᛁᚷᛖᛒᛁᛞᛞᚪᚦ᛫ ᚦᚫᚱ᛬ ᛋᚪᚢᛚᛖ
jilsuiþ: arærde: æfte berhtsuiþe· bekun onbergigebi*dd*aþ· þær: saule

If this text were encoded with a ZERO-WIDTH LIGATOR, all the user would have to do is ensure that ZERO-WIDTH LIGATOR appears between ᛞ and ᛞ ("&" is used here to show the ZERO-WIDTH LIGATOR):

ᛄᛁᛚᛋᚢ'ᚦ᛬ ᚪᚱᚫᚱᛞᛖ᛬ ᚫᚠᛏᛖ ᛒᛖᚱᚻᛏᛋᚢ'ᚦᛖ᛫ ᛒᛖᛣᚢᚾ ᚩᚾᛒᛖᚱᚷᛁᚷᛖᛒᛁᛞ&ᛞᚪᚦ᛫ ᚦᚫᚱ᛬ ᛋᚪᚢᛚᛖ

If on the other hand there were no ZERO-WIDTH LIGATOR, but instead a ZERO-WIDTH NON-LIGATOR were implemented to prevent ligatures where they were activated globally, this text would have to be coded thus ("∅" is used here to show the ZERO-WIDTH NON-LIGATOR):

ᛄᛁᛚᛋᚢ'ᚦ᛬ ᚪᚱᚫᚱᛞ∅ᛖ᛬ ᚫ∅ᚠᛏᛖ ᛒᛖ∅ᚱᚻ∅ᛏᛋᚢ'ᚦ∅ᛖ᛫ ᛒ∅ᛖᛣ∅ᚢ∅ᚾ ᚩᚾᛒᛖ∅ᚱᚷᛁᚷᛖ∅ᛒᛁᛞᛞ∅ᚪᚦ᛫ ᚦᚫ∅ᚱ᛬ ᛋᚪ∅ᚢᛚ∅ᛖ

The 55-character text would have to be burdened with 14 ZERO-WIDTH NON-LIGATORS to override an automatic ligation option; otherwise it would display with all the potential ligatures in this particular font (note the ᛞᛞᚪ *dda* triplet):

ᛄᛁᛚᛋᚢ'ᚦ᛬ ᚪᚱᚫᚱᛞᛖ᛬ ᚫᚠᛏᛖ ᛒᛖᚱᚻᛏᛋᚢ'ᚦᛖ᛫ ᛒᛖᛣᚢᚾ ᚩᚾᛒᛖᚱᚷᛁᚷᛖᛒᛁᛞᛞᚪᚦ᛫ ᚦᚫᚱ᛬ ᛋᚪᚢᛚᛖ

It should be noted that Runic ligation must be considered now to be productive; a web search on "bind-runes" will lead to a large number of sites explaining how runes can be combined by the modern user for various "magical" purposes. More serious, however, would be the use runologists might make of sorting and searching bind-rune for statistical analyses, etc. – which currently cannot be done effectively, as bind-runes can only be achieved at present either by private-use characters or markup. Plain text character coding is *certainly* more reliable.

9. Examples of Gaelic typographic ligatures. Two identical texts (Luke 2: 8–14) from two different publications using the same typeface but in which different ligature choices were made. One might call this "ligature classment". Both samples are from Uilliam O'Domhnuill's translation of the New Testament: the sample on the right was published in 1827 (173 years ago) in Dublin by G. and J. Grierson and M. Keene for the British and Foreign Bible Society; the sample on the left was published in 1849 (151 years ago) in Dublin by Hardy and Sons. Words containing typographic ligatures are shown in outline style.

8 ¶ Agus do ḃádar áoḋairiḋe sa dúiṫċe sin na gcóṁnuiḋe amuiġ, 7 ag deanaṁ faire óiḋċe air a dtréud.
9 Agus, féuċ, do ṡeas aingeal an Tíġearna láiṁ léo, agus do ṡoillsiġ glóir an Tíġearna na dtimċeall: 7 do ġab eagla mór íad.
10 Agus a dúḃairt an taingeal riú, Na bíoḋ eagla oruiḃ: óir, féuċ, a táim ag soisgéuluġaḋ díḃ gáirdeaċus mór, noċ bías do gaċ uile ṗobal.
11 Oír ruġaḋ an Slánuiġṫeóir ḋáoiḃ a niúġ (éaḋon Críosd an Tíġearna), a gcaṫruiġ Dháibi.
12 Agus ag so cóṁarṫa díḃ; Do ġéubṫáoi an leanḃ ceanguilte a ngiobluiḃ, na luiġe a maiñséur.
13 Agus do bí go hobañ cuideaċda ṁór do ṡlúaġ neaṁḋa a bfoċair an aingil ag molaḋ Dé, 7 ag ráḋ,
14 Glóir do Dhía añ sna hárduiḃ, 7 síoḋċáin air an dtalaṁ, deaġṫoil do na dáoiniḃ.

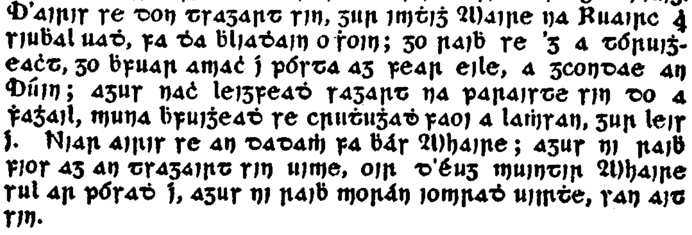
8 ¶ Agus do ḃádar áoḋairiḋe sa dúiṫċe sin 'na gcóṁnuiḋe amuiġ, 7 ag déanaḋ faire óiḋċe air a dtréud.
9 Agus, feuċ, do ṡeas aingeal an Tíġearna láiṁ leo, 7 do ṡoillsiġ glóir an Tíġearna 'na dtimċeall: 7 do ġab eagla ṁór iad.
10 Agus a dúḃairt an taingeal riu, Na bíoḋ eagla oruiḃ: óir, feuċ, a táim ag soisgeuluġaḋ díḃ gáirdeaċus mór, noċ bías do gaċ uile ṗobal.
11 ´Oir ruġaḋ an Slánuiġṫeoir ḋáoiḃ a niuġ (éaḋon Críosd an Tíġearna), a gcáṫruiġ Ḋáibi.
12 Agus ag so cóṁarṫa díḃ; Do ġéabṫaoi an leanḃ ceanguilte a ngiobluiḃ, 'na luiġe a maiñséur.
13 Agus do bí go hobañ cuideaċda ṁór do ṡlúaġ neaṁḋa a bfoċair an aingil, ag molaḋ Dé, 7 ag ráḋ,
14 Glóir do Ḋía añsna hárduiḃ, 7 síoḋċáin air an dtalaṁ, deaġṫoil do na dáoiniḃ.

To facilitate comparison, the texts are given below in Roman type with ligatures expanded, with the words containing the expanded ligatures printed in outline style.

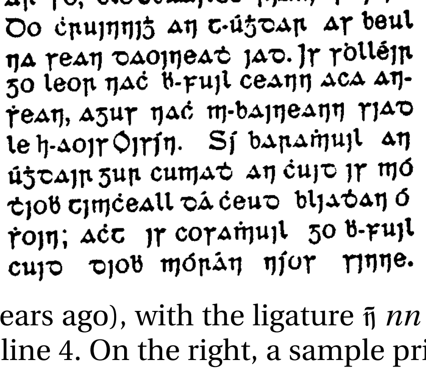
8 ¶ Agus do ḃádar áodairiḋe sa dúiꞇċe sin na gcóṁnuiḋe amúiġ, ⁊ ag deanaṁ ꝼaire óidċe air a dꞇréud.	8 ¶ Agus do ḃádar áodairiḋe sa dúiꞇċe sin ’na gcóṁnuiḋe amuíġ, ⁊ ag deanaḋ ꝼaire óidċe air a dꞇréud.
9 Agus, ꝼéuċ, do ṡeas aingeal an Tíġearna láiṁ léo, agus do ṡoillsiġ glóir an Tíġearna na dꞇimċeall: ⁊ do ġaḃ eagla mór íad.	9 Agus, ꝼeuċ, do ṡeas aingeal an Tíġearna láiṁ léo, agus do ṡoillsiġ glóir an Tíġearna na dꞇimċeall: ⁊ do ġaḃ eagla mór íad.
10 Agus a dúḃairꞇ an ꞇaingeal riú, Na bíoḋ eagla oruiḃ: óir, ꝼéuċ, a ꞇáim ag soisgéuluġaḋ ḋíḃ gáirdeaċus mór, noċ ḃías do gaċ uile ṗobal.	10 Agus a dúḃairꞇ an ꞇaingeal riú, Na bíoḋ eagla oruiḃ: óir, ꝼeuċ, a ꞇáim ag soisgéuluġaḋ ḋíḃ gáirdeaċus mór, noċ ḃías do gaċ uile ṗobal.
11 Oír rugaḋ an Slánuiġꞇeóir ḋáoiḃ a niúġ (éaḋon Críosd an Tíġearna), a gcaꞇruiġ Dháiḃi.	11 ´Oir rugaḋ an Slánuiġꞇeóir ḋáoiḃ a niúġ (éaḋon Críosd an Tíġearna), a gcaꞇruiġ Ḋáiḃi.
12 Agus ag so cóṁarꞇa ḋíḃ; Do ġéuḃꞇáoi an leanḃ ceanguilꞇe a ngiobluiḃ, na lúiġe a mainnséur.	12 Agus ag so cóṁarꞇa ḋíḃ; Do ġéuḃꞇáoi an leanḃ ceanguilꞇe a ngiobluiḃ, na lúiġe a mainnséur.
13 Agus do ḃí go hobann cuideaċda ṁór do ṡlúaġ neaṁḋa a bꝼoċair an aingil ag molaḋ Dé, ⁊ ag ráḋ,	13 Agus do ḃí go hobann cuideaċda ṁór do ṡlúaġ neaṁḋa a bꝼoċair an aingil ag molaḋ Dé, ⁊ ag ráḋ,
14 Glóir do Dhía ann sna hárduiḃ, ⁊ síoḋċáin air an dꞇalaṁ, deaġꞇoil do na dáoiniḃ.	14 Glóir do Dhía annsna hárduiḃ, ⁊ síoḋċáin air an dꞇalaṁ, deaġꞇoil do na dáoiniḃ.

Although the *same* font is used for these examples, ligature behaviour is not the same for the two texts, and indeed in the first text, *eagla* ‘fear’ is written both eagla (verse 9) and ęgla (verse 10), and *air* ‘on’ is written both ą (verse 8) and air (verse 14). The glyph selection is not rule-governed, and it is impractical at best to suggest that the font designer, even though he or she is *permitted* to specify various “classes” according to the AAT/OpenType model, should be *able* to specify all the possible glyph-selection levels in a particular font, because the choice may differ greatly from document to document. It is not as it is in English, where one might have simplistic choices like “all *f*-ligatures”, “all *f*-ligatures and *ct*/*st* ligatures”, and “all available ligatures”. In scripts like Runic, Ogham, and Old Hungarian, where nonce-ligatures often occur (and where there are also examples like eagla/ęgla), the ZERO-WIDTH NON-LIGATOR would be a serious nuisance.

There are also a few other minor spelling differences between the two texts: *Óir* ‘for’ is written Oír and ´Oir in verse 11; also ꝼéuċ and ꝼeuċ for *féaċ*; also Dhía/Ḋía for *Ḋia*. The specific *Ó* = *´O* = *Oѷ* orthographic conventions used in a printed text may be of interest to researchers sorting or searching – syntactic analyses of the word *óir* ‘for, because’ (French ‘*car*’) require that the researcher knows that the spellings *´oir* and *oír* also apply. Note in verse 8 the word *amuiġ* ‘outside’ appears as amǘġ (with ligature, = amúiġ) and amuiġ. Implementation of ZERO-WIDTH LIGATOR would code the first as amú&iġ. Could researchers wish to distinguish amǘġ/amúiġ/amuiġ for some purpose? With ZERO-WIDTH LIGATOR, they would be able to do so, more simply than with markup.



Dainṁḋeóin sin ċuaiḋ an Seoirseaċ fa ḋéin añ tragairt a ṗós é féin is Maire san áit a rugaḋ ’s a beaṫuġaḋ í. Bhí sin deiċ míle fiṫċead on áit a mbiaḋ siad na gcoṁnaiġ. D’airis se don tragart sin, gur imṫiġ Maire na Ruairc ą siubal uaḋ, fa ḋa bliaḋain o ṡoin; go raib se ’g a tóruiġeaċt, go bfuair amaċ i pórta ag fear eile, a gconḋae an Dúin; agus naċ leigfeaḋ sagart na paraiste sin do a ḟaġail, muna bfuiġeaḋ se cruṫuġaḋ faoi a laiṁsan, gur leis í. Niar airis se an dadaṁ fa bás Mhaire; agus ni raib fios ag an tragairt sin uime, oir d’éug muintir Mhaire sul ar pósaḋ í, agus ni raib morán iomraḋ uirṫe, san ait sin.



Is cosaṁuil naċ raib aon ċeann do na sgeultaib atá ins an leabar so, cloṫbuailte riaṁ roiṁe. Do ċruinniġ an t-úġdar ar beul na sean daoineaḋ iad. Is soilléir go leor naċ b-fuil ceann aca anṡean, agus naċ m-baineann siad le h-aois Oisín. Sí baraṁuil an úġdair gur cumaḋ an ċuid is mó ḋiob timċeall dá ċeud bliaḋan ó ṡoin; aċt is cosaṁuil go b-fuil cuid diob mórán níos sinne.

On the left, a sample of text set in Watts type in 1845 (155 years ago), with the ligature ñ *nn* used once in line 1 and the ligature ą *air* used once at the end of line 4. On the right, a sample printed in the *Irish Echo* in America in 1890 (110 years ago), which uses no ligatures at all.

The AAT/OpenType model as currently applied to Latin seems to have been developed only to cater for *decorative* ligation, but as we have seen, ligation may not be simply decorative. Editorial choices regarding spacing, legibility, and so on were made in preparing the texts above. Such choices may be considered significant in textual analysis; certainly as noted above the presence of bind-runes in Runic texts is likely to be considered significant by runologists. Use of ZERO-WIDTH LIGATOR to effect typographic ligation would permit researchers to sort and search taking ligatures into account, *because they would be explicitly marked in plain text by the encoding.*

Other forms of ligation vs. non-ligation are significant as well: in ISO/IEC JTC1/SC2/WG2 N2141, I noted that

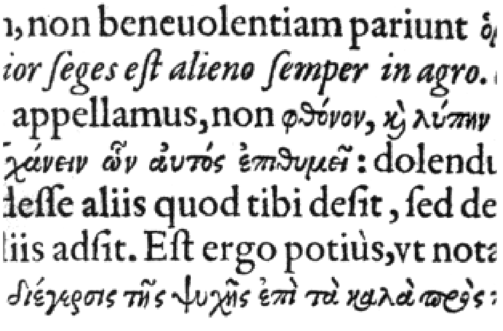
> For most languages, Latin *fi* and *fl* ligatures are not optional in *good typography* (Turkish and Azerbaijani require a distinction to be made between *fi* and *fi* but presumably permit *fl* ligatures.) … The end-user probably cannot [reprogram the ligature management mechanism to override automatic font ligation]. But he or she can choose to insert or delete a ZERO-WIDTH LIGATOR. An “unintended nasty consequence” could, for instance, make a font unusable for Turkish or Azerbaijani – a problem avoided by the use or non-use of ZERO-WIDTH LIGATOR.

Ligatures of *fi* (*fi*) are proscribed in Turkic Latin environments. The current model puts the burden of providing a specific “Turkic” level of ligation into their fonts upon the designers. How many designers are linguistically literate enough to anticipate this? The number is certainly a very small subset of font designers. With ZERO-WIDTH LIGATOR, a language-independent and script-independent solution, users will be able to select, manually or automatically depending on inputting software, the ligatures required, and the need to have fonts containing an explicit Turkic option does not arise, as the ZERO-WIDTH LIGATOR solution is general at the level of coding.

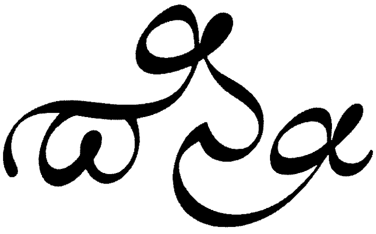
> NOTE: Being conservative, and keeping to 16 Turkic languages which could make use of the Turkish *i*/*ı* distinction, and bearing in mind the fact that some of these populations presently use the Arabic or Cyrillic scripts, the user community potentially requiring Turkic handling of the *fi* ligature is approximately 105,750,000. Of these, we know that 62% of them (the Turks and the Azeris) are in fact writing with the *i*/*ı* distinction. That’s 66 million users.

In the discussions to date, we have seen an attachment by aficionados of the AAT/OpenType model to its purported richness, but failing the introduction of a ZERO-WIDTH LIGATOR which selects glyphs based on coded character triplets, the end user is stuck with whatever tables a designer has seen fit to include in his font, which could well render a great many Latin fonts unusable to up to a hundred million Turkic-speaking users. Yet the UCS already permits coded-character-sequence based glyph selection for Brahmic scripts; it is illogical that we not do so the same for European scripts.

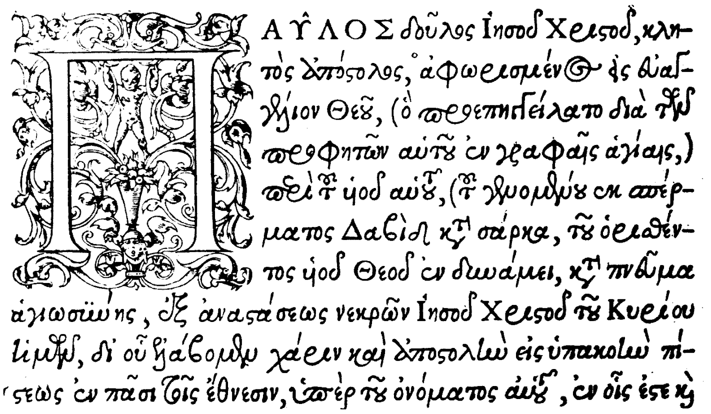
10. Examples of Greek typographic ligatures. Early Greek typography makes use of a very rich set of ligatures.



, non beneuolentiam pariunt
ior ſeges eſt alieno ſemper in agro.
appellamus, non φθόνον, καὶ λύπην
χάνειν ὧν αὐτὸς ἐπιθυμεῖ: dolend
deſſe aliis quod tibi deſit, ſed de
iis adſit. Eſt ergo potiùs, vt not
διέγερσις τῆς ψυχῆς ἐπὶ τὰ καλὰ πρὸς:

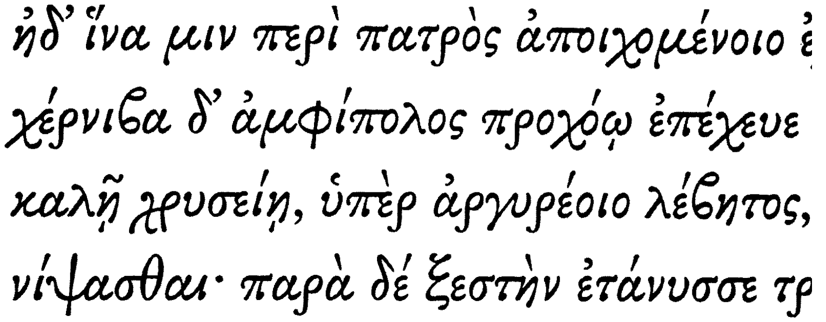


On the left, a text in Latin and Greek, both making use of typographic ligatures (ca. 1770, 230 years ago). On the right, a ligature for the word παϱα ‘near’.



ΠΑΥΛΟΣ δούλος Ιησου Χϱιστου, ϰλητός απόστολος, αφωϱισμένος εις ευαγγέλιον Θεού, (ό πϱοεπηγγείλατο διά των πϱοφητών αυτού εν γϱαφαίς αγίαις,) πεϱί του υιου αυτού, (του γενομένου εϰ σπέϱματος Δαβίδ ϰατα σάϱϰα, τού οϱισθέντος υιου Θεου εν δυνάμει ϰατα πνεύμα αγιωσύνης εξ αναστάσεως νεϰϱών Ιησου Χϱιστου τού Κυϱίου ημων, δι ού ελάβομεν χάϱιν ϰαί αποστολήν εις υπαϰοήν πίστεως εν πάσιν τοίς έθνεσιν, υπέϱ τού ονόματος αυτού, εν οίς εστε ϰαί.…

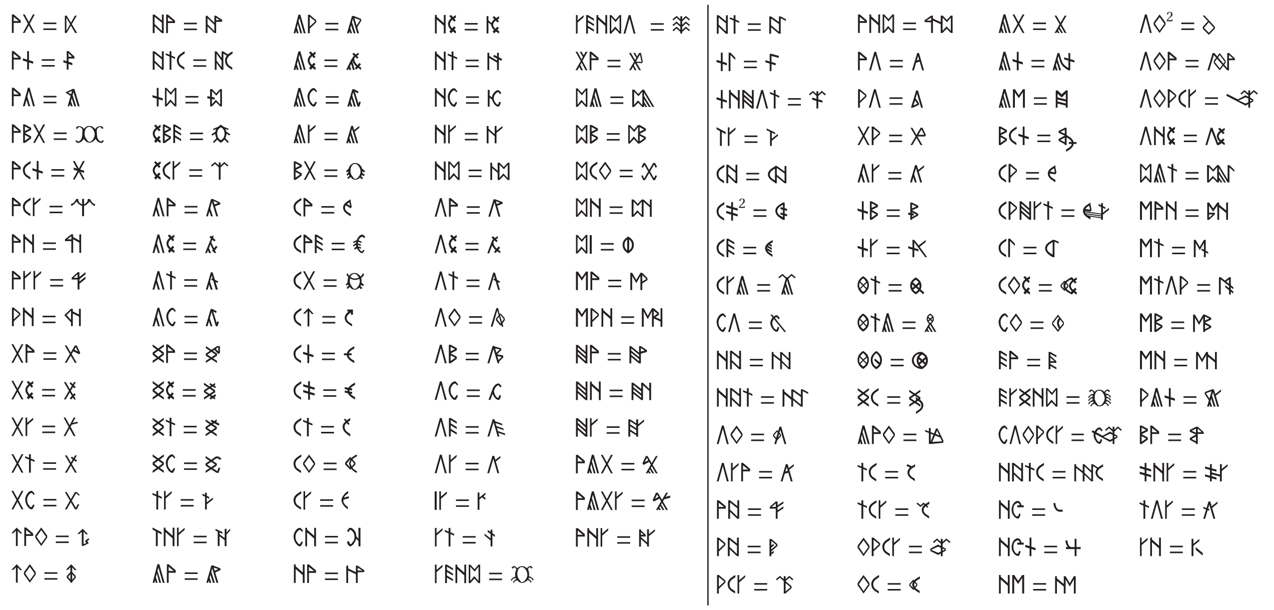
On the left, text printed by Robert Estienne in a font by Claude Garamond, Paris 1550 (450 years ago), showing an extremely complex array of ligatures; on the right, the same text (Romans 1: 1–6) transcribed. Which is simpler, using markup to represent the text, or using the ZERO-WIDTH LIGATOR? Setting this text would be time-consuming either way – but the ZERO-WIDTH LIGATOR is portable in plain text.



ἠδ’ ἵνα μιν περὶ πατρὸς ἀποιχομένοιο ἐ
χέρνιβα δ’ ἀμφίπολος προχόῳ ἐπέχευε
καλῇ χρυσείῃ, ὑπὲρ ἀργυρέοιο λέβητος,
νίψασθαι· παρὰ δὲ ξεστὴν ἐτάνυσσε τρ

Modern font revival of Alexander Wilson’s 1756 Greek typeface, with 17 tying ligatures.

11. Ligatures in Old Hungarian. Old Hungarian is perhaps one of the best examples with regard to this discussion. The principal reason that Old Hungarian has not already been accepted for coding in the UCS is the ligation requirement. The basic set consists of 45 letters, a set of 79 “canonical” ligatures promulgated by Hungarian experts on the net, and a set of 63 ligatures which I had found in two other sources. The “official” expert view is that the 63 additional ligatures are “unsafe” (whatever that means) and that only the 79 should be supported. It is my belief that one of the reasons for this is the space limitation of the currently-available PC implementations of Old Hungarian fonts. I can find no principled reason for not accepting these additional ligatures as authentic ligatures (found as they are in texts expounding the use of ligatures in the Old Hungarian script). Indeed, the Hungarian experts want to insist that the 79 “canonical” ligatures be explicitly encoded in the UCS, and have not accepted my assertions that the character/glyph model supports the Old Hungarian ligation requirements. In part, this is because of the vague nature of the current model, which would have ligatures either on or off; I cannot tell the Hungarians “no problem, in plain text you can just use ZERO-WIDTH LIGATOR and you will be fine”, but instead have to refer them to as-yet-unavailable AAT/OpenType technologies, which, as presently constituted, offer no plain text mechanism for ligature presentation, which the Hungarians don’t like. It is for this reason that the Hungarian experts insist on hard coding of their ligatures *even in the UCS*, so that they are available when required. ZERO-WIDTH LIGATOR would guarantee that whenever required, a ligature could be coded, while leaving ordinary representation available for general purposes. In fact, of the 79 “safe” ligatures, six of them are not even considered by these experts as ligatures, but rather are viewed as Devanagari क्ष *kṣa* sometimes is, as unique indivisible characters, analysed by shape only. (The experts even give dubious “Egyptian” scarab-beetle etymologies to explain these “bug signs”.) All of these have normal ligature transliteration decompositions, and if ZERO-WIDTH LIGATOR were available, it would be far easier to assure the Hungarian experts that 𐳀 + ZWL + 𐳘 + ZWL + 𐳂 would be used to represent 𐳃 *amb* with concomitant support for lexical, sorting, and searching operations. Of the 63 "unsafe" ligatures, the Old Hungarian data shows the extreme example of *Oskánt*. But this is no different from Devanagari र्ष्ट्यां *rṣṭyāṁ* where the sequences 𐳛 + ZWL + 𐳤 + ZWL + 𐳓 + ZWL + 𐳁 + ZWL + 𐳙 + ZWL + 𐳦 and र + ् + ष + ् + ट + ् + य + ा + ं can each point to unique glyph cells in a font.



12. Rules for ligation in Fraktur German. From Albert Kapr, *Fraktur: Form und Geschichte der gebrochenen Schriften.* Mainz: Hermann Schmidt. 1993. ISBN 3-87439-260-0. My translation.

2. Fraktur ligatures: ch ck ff fi fl ft ll ſch ſi ſſ ß ſt tt tz (ch ck ff fi fl ft ll ſch ſi ſſ ß ſt tt tz)

The summarized rules of use (ignoring the case of the commonly known ß):

— Ligatures are used if they do not disrupt linguistic correctness: doch, Ruck, hoffen, verfilzt, flach, oft, lallen, [Fiſch], ſie, Waſſer, Aſt, retten, Katze (doch, Ruck, hoffen, verfilzt, flach, oft, lallen, [Fiſch], ſie, Waſſer, Aſt, retten, Katze)

— They are not employed in word compounds: auffordern, Schilfinſel, Schnupftuch (auffordern, Schilfinſel, Schnupftuch)

— Word stems must be observed: ich kaufte (ich kaufte), but single letters at the ends of words are not 'detached': gekauft, Aufl. (gekauft, Aufl.)

— Syllables are not relevant: ſaftig (ſaftig) is set with a ligature; the word stem is 'Saft'.

— Should three letters which appear together, of which a ligature could be made from any two of them, the syllable-division decides: pfiffig, Offizier, Souffleur (pfiffig, Offizier, Souffleur)

— Ligatures are not used to save space. Exceptions: ch, ck, tz and ß (ch, ck, tz and ß)

There are apparently no ligature triples in Fraktur. Note the recently discussed example Wachstube *Wachstube* 'wax tube' vs. Wachſtube *Wachſtube* 'guardroom'.

13. Costs of ZERO-WIDTH LIGATOR implementation. In the short term, it is true that the provision of a ZERO-WIDTH LIGATOR will not solve all problems immediately, and it does (or could) necessitate a reanalysis of recommended practice with regard to the current AAT/OpenType model. At the same time, however, it must be noted that the takeup of that model has been less than admirable, so we are not talking about a huge change in coding practice which will costs millions to a great many companies. To date I have been informed of only one product that implements the current AAT/OpenType model, and I am unconvinced that this is the best model given the ligation requirements of European scripts (which are a good deal greater than the use of *f*-ligatures in English and German).

I argue that a single consistent mechanism for ligature formation will benefit users of the UCS, that this is already present in the coding conventions for Brahmic scripts, and that the greatest number of user needs can be satisfied by harmonizing European ligature selection with the proven-successful mechanism used for Brahmic scripts.

John Jenkins asked me what mechanism I would employ to achieve ligature *variants* but I say that the same mechanism used for Devanagari *et al.* will serve here. क + ् + ष yields क्‍ष, but it may also yield क्ष. The same sort of thing happens in Tibetan. Mechanisms exist in Devanagari WorldScript and Tibetan (third-party) WorldScript to allow the user to select *variant* ligatures – but the underlying encoding (used for sorting and searching) specifying *that* a ligature is used remains the same, and whatever mechanism is used for those scripts should serve just as well for European scripts. The fact is that the current Brahmic model works perfectly well, and is superior to the ad-hoc AAT/OpenType ligature model, because it is consistent with *established* UCS principles of using coded character sequences to specify glyph selection. There is no need to make European scripts a non-plain-text exception, since ligation behaviour in European scripts, while less extensive than it is in Brahmic, can easily benefit from the use of the same principles, if we will only choose, sensibly, to normalize *now*.

Use of ZERO-WIDTH LIGATOR makes glyph selection simpler and application-independent, since applications will have to deal with doublets and triplets of base-character + combining-character sequences in any case – ZERO-WIDTH LIGATOR simply adds a new set of coded-character-to-string vectors.

Practically speaking, with regard to cursor placement, etc., implementation of ZERO-WIDTH LIGATOR is no different from implementation of SOFT HYPHEN. With XPress now, when selecting text containing the SOFT HYPHEN, I often meet with the invisible SOFT HYPHEN character between two other characters in a word. It took me only a moment to learn that this occurred, and that was years ago; it is second nature to me now.

13. Excerpted discussion of the ZERO-WIDTH LIGATOR from the Unicode list, with comments.

John Jenkins said:

> I haven't seen anyone claim that ligation control should be handled entirely by software. The OpenType/AAT model is explicitly opposed to that assumption.

Well, that is a fault in the analysis that led to the present AAT/OpenType model. It works for trivial substitutions like standard English *f*-ligature behaviour. It fails in the long term where more complex ligation behaviour is indicated. But it's not a total failure – because the AAT/OpenType model *already* supports the basic model of Brahmic character-based glyph selection. The failure is insisting on trying to do Latin ligation with a *different* model, *i.e.*, without explicit encoding. Abandon *this* and you are left with something that works in *all* contexts.

> The end user still has complete control over the process. The software merely enables this and provides default behavior.

But the choice and formatting of the user (*i.e.* the specific ligature behaviour selection) is not preserved in plain text. This is another fault that can be easily abandoned.

> It would be possible to write such an editor for Mac OS 9 in about a page of code. We're working on sample code and demo applications to show how this can be done. And it's been possible to do ever since we released the late lamented GX five years ago.

When will such an editor be available? It wasn't available 5 years ago either.

> Adobe InDesign – and admittedly non-cheap program – can handle an awful lot of Latin ligatures. I believe it could handle anything you put in your OpenType font, at least manually. Given InDesign's extensive plug-in architecture, it would be possible to get it to do whatever you want.

If, and only if, you are a *computer programmer* and can write a plug-in. Most users and font designers are *not* programmers. Even expert users like me are *not* able to do this. ZERO-WIDTH LIGATOR puts real control at the fingertips of the end user, and facilitates the vectoring of glyphs for the designer. Surely this benefits everyone. The font designer and the end user are the people who have to deal with the provision and use of ligatures in actual text. The big companies have already done the right thing for Brahmic scripts, and now only need to accept a single logical model and employ the same model for Latin that we promulgate to everyone else.

> There are two issues here. One is getting system software support. The other is getting applications to take advantage of the system software support. The latter can be an enormous uphill battle, as our experience getting people to support ATSUI shows.

System software that supports the Brahmic paradigm should be able to be configured to implement ZERO-WIDTH LIGATOR with a minimum of effort and expense.

> The former is also enormously problematical. The problem is that the TrueType spec doesn't offer any direct support for mapping multiple characters to single glyphs – the presumption is that this is handled in the AAT/OpenType tables. I don't know how OpenType libraries like UniScribe or CoolType work, but I know enough about the guts of ATSUI to say that it would be fairly difficult to get it to handle ZERO-WIDTH LIGATOR;

Not at all. If *e* + ◌́ = *é*, why shouldn't *f* + ZWL + *l* = *fl*?

> ZERO-WIDTH NON-LIGATOR would be comparatively simple. I would imagine that OpenType would have similar problems. Basically ZERO-WIDTH LIGATOR would be useless except for a plain-text exchange mechanism, and even there it would be problematical.

It is a question of ensuring that certain strings of characters point to certain glyph cells. You guys have been telling us font designers for *years* that smart fonts is the way to go. Well, I for one believed you, and I am telling you that I can't get what I need from your present model, but with a simpler model (already compatible with AAT/OpenType as applied to Brahmic) we have no problems at all, apart from a short-term model overlap.

> I think it's a bit on the naïve side to assume that just adding ZERO-WIDTH LIGATOR would solve the problem. We'd be three or four years at best before we'd get support at the OS level and then down to the app level. The point is that there is already a mechanism in place on both Windows and the Mac that can solve the problem. We'd be far better off IMHO to define exchange formats for cross-platform use that includes ligation information.

I am aware of the temporal problem. I can live with it; the UCS is forever. I am, however, convinced that ZERO-WIDTH LIGATOR is the simplest and best solution. Your "mechanism in place on the Mac" is *not* in place on the Mac, except in a technology which has *not* been snapped up by hardly anyone. Sorry, gods know I don't like to say this, but there is *no* mechanism for this on my Mac, and the mechanism that you are promulgating *does not serve my needs.* It is needlessly complex and *too* open-ended. It burdens me, if I want to make and Old Church Slavonic font that handles all kinds of OCS typography, with knowing what all the possible permutations of ligatures could be, but this is is *impossible* to know, since the permutations can differ from document to document and even from sentence to sentence within a document. What I *need* is a way to *type* what I need and to have the data preserved in plain text. You can automate that input to your heart's content for Latin *f*-ligatures. Just give us a consistent *coding* model for ligature behaviour for European and Brahmic scripts alike. Bite the bullet – this problem will *not* go away till you do what we tell you we need! The problem has been identified and discussed for a decade already. The AAT/OpenType model has *not* been implemented, and it is *inadequate.* A simpler model, compatible with AAT/OpenType and current Brahmic practice, exists, and if you implement it you will serve the user. You need to *stop* trying to cheat the rendering engine by pretending that Latin *f*-ligatures can be handled without character coding. Why do we not handle non-standard Arabic glyph behaviour with markup, instead of with an explicit ZERO-WIDTH JOINER and ZERO-WIDTH JOINER? Your (*pl.*) arguments are just as valid for that situation as they are for Latin ligature behaviour. Either character-code sequence points to glyph cell, or that kind of selection should *always* be handled by markup. The mixed model is inadequate and problematic. Character-code sequence is by far and away more preferable.

¶

Peter Constable said:

> But ligatures that are in free distribution with non-ligated forms are exactly the kind of thing that ought to be specified using style/formatting information applied to the run of characters: it does *not* have anything to do with the meaning of the text but is purely there for presentation purposes.

Not so: ZERO-WIDTH JOINER/ZERO-WIDTH NON-JOINER also have no relation to the *meaning* of coded text string – they only deal with text *presentation*.

> Ligation in this case *can* be handled using such a mechanism, selected by the user using some UI device, with the OS/rendering engine applying a feature that applies the appropriate substitution. In such cases of ligation, it should not be a requirement that the ligature be retained when exchanged via plain text any more than should the choice of typeface, point size, bold, italic, etc. Such presentation information does not belong in plain text.

No, because italic markup switches *between* fonts (as opposed to switching *typeface*), and ligation selects specific cells *within* a font.

> If we start talking about encoding in Unicode all presentation distinctions in ancient documents that might prove to be significant (but also might not), won't we end up turning this character encoding standard into a glyph encoding standard? Maybe this rhetorical question is reactionary and that is could be feasible to add such to Unicode in a controlled manner. Just sounds scary.

My 1849 Irish bible is not "ancient". Anyway, the ZERO-WIDTH LIGATOR is just a way of doing something we *already* do in the standard in order to effect an *identical* kind of glyph selection behaviour with Brahmic scripts and fonts.

¶

Asmus Freytag said:

> What is at the heart of this recurring request is that support for many scripts (or older typographies) is incomplete without an *interchangeable* method of indicating the precesence or absence of ligatures.

Correct.

> Posing this question allows us to consider the full-featured typorgraphic and aesthetic requirements for ligation – as well as any inherent regularities. Once we have a design in place for interchanging ligatures with marked up text, we can revisit that and see whether replacing markup instructions by character codes gives better results.

This is a waste of time. We *already* have a working model with VIRAMA, and ZERO-WIDTH LIGATOR is analogous to that and its implementation would make the logic for font implementation for all these scripts the *same*.

> Both concepts ('ligate here', 'don't ligate here') can in principle be expressed with HTML or XML style markup – I have seen too little discussion of what this markup should be like, and what the consequences are of it being present in the middle of words. Is that something that the HTML/XML community wants to deal with?

Doing this would be just as ad-hoc as the current AAT/OpenType model is, for European scripts, and all we are talking about is character-based glyph cell selection.

> The strongest arguments in favor of character codes come from those who have for long time needed to 'trick' various applications into supporting languages that they were not explicitly designed for. If character codes would result in 'enabling' many of these implementations, by letting the author communicate with the rendering engine, so to speak, that is itself a valid argument to consider. (It would need some actual case studies where this approach is shown to work).

I performed such "tricks" in order to prepare the present document.

> Still, even that would need to be contrasted with the cost to applications that do not know about these as characters and end up showing ‘boxes’.

I could live with this in the short term. The long-term advantages to the designer and the user *far* outweigh this minor inconvenience.

¶

Gary Roberts said:

> Yes. There is definitely an issue of how to accomplish what one wants in a way that will be implemented. For example, if the solution relies on language tags (*e.g.* dictionary based solutions), then it is of little use if companies don’t provide support for your language.

Which is, of course, unacceptable, given the number of the world’s languages.

> On the other hand, the SOFT HYPHEN is generally implemented, and supports languages that haven’t even been invented yet. Now, one could argue whether SOFT HYPHEN is best implemented as markup or as the addition of a new character. I tend to read and create markup files by hand. My tendency is to prefer markup when there is some span to the markup. The more characters the markup is likely to affect, the more I prefer it to adding a character. SOFT HYPHEN is an example where there is no span at all, and it makes sense to solve the issue with a SOFT HYPHEN character. I see ZERO-WIDTH LIGATOR as a substitute for markup having a span of two or three characters, which still makes it attractive as a new character solution. It also seems more flexible.

Yes! As I said above, markup could substitute for ZERO-WIDTH JOINER and ZERO-WIDTH NON-JOINER just as “easily”. But it would be absurd to do so.

> Say that I often deal with fonts that have only ligature pairs, given the choice of *ffi* or *ffi*, I always prefer *ffi*, but my colleague prefers *ffi*. We both prefer *ffi* as a single ligature if it exists in the font. What markup gives each of us the results we prefer? For &=ZERO-WIDTH LIGATOR, the answer is f&fi for me, and ff&i for my colleague. Note that ZERO-WIDTH NON-LIGATOR is not useful for this case.

I agree, ZERO-WIDTH NON-LIGATOR is not useful here, but the solution is simple:

1. For an encoded sequence f&f&i, if <ffi> exists, print it.
2. If <ffi> does not exist, but <ff> exists, print it and ignore the second &.
3. If <ffi> and <ff> do not exist, but <fi> exists, print it and ignore the first &.
4. If <ffi> and <ff> and <fi> do not exist, ignore both &s.
5. If <ffi> does not exist, but <ff> and <fi> exist, and the user prefers f<fi> to <ff>i, user deletes the first &.

> Search engines that are not markup savvy (including my eyes) find markup more disruptive than a single character, particularly if it occurs frequently.
>
> Search en-gines that are not mark-up sav-vy (in-clud-ing my eyes) find mark-up more dis-rup-tive than a sin-gle char-act-er, par-tic-u-lar-ly if it occ-urs fre-quent-ly.
>
> Search en<SH>gines that are not mark<SH>up sav<SH>vy (in<SH>clud<SH>ing my eyes) find mark<SH>up more dis<SH>rup<SH>tive than a sin<SH>gle char<SH>act<SH>er, par<SH>tic<SH>u<SH>lar<SH>ly if it occ<SH>urs fre<SH>quent<SH>ly.
>
> Note that I have chosen relatively non-intrusive markup.

The arguments above show that we should not be using markup for ligation because the character-sequence-to-glyph model is already there and should be preferred. But Gary’s example is good, especially in the “Show Invisibles” option that good applications offer: suf&f&icient would be friendlier than su<LIG>ffi</LIG>cient in contexts where invisible characters and markup are displayed to the user.

¶

Asmus Freytag said:

> The beauty of character codes as controlling agents lies in the fact that they directly interact with the ligating engine's context processing. As has been pointed out on the Unicode list, f<ZWL>f is supported immediately, as long as dumb fonts or unrelated processes can handle it as a zero-width character (ignorable).

Only in a situation which has a meta-markup turning on one (or more) levels of ligature class. This mixes markup with coding for the same operation (glyph selection) and is not consistent with VIRAMA/ZERO-WIDTH JOINER conventions.

¶

Mark Davis said:

> With ZERO-WIDTH LIGATOR, the font designer has to actively add *all* sequences of the form:
>
> "f" + ZERO-WIDTH LIGATOR + "i" -> "fi" ligature
> "f" + ZERO-WIDTH LIGATOR + "f" -> "ff" ligature
> ...
> if those ligatures are to work with ZERO-WIDTH LIGATOR. There is no magic wand that tells the software how ZERO-WIDTH LIGATOR would work, otherwise.

Yes. The same principles we use for sequences of base and combining characters apply. For the user, this could be handled by input, as it is (at least minimally) in XPress. For the designer, he or she has to do this for all precomposed glyphs of combining characters anyway (at least for precomposed characters which are post-normalization). What's the difference? ZERO-WIDTH LIGATOR makes the designer's job easier: he or she knows that certain coded-character sequences will select the vectored glyphs; this is the best use of the smart font model.

¶

Lloyd Anderson said:

> [Mark Davis' model above] is not the context I was envisaging at all. My understanding was that those concerned with the Latin alphabet would have fonts containing "f" + "i" -> "fi" ligature (and so on), and a global ligature-on setting would cause these rendering glyphs to be used, while a global ligature-off setting would cause these rendering glyphs to not be used.

The problem is that the global ligature on/off model mixes markup with character coding.

> There is no statement that I have seen made by Everson that would require throwing out such an implementation.

It would be cleaner if *fi*/*fl* ligatures were *also* encoded with ZERO-WIDTH LIGATOR. Users are now presented in some (few) applications like XPress with an option to use these ligatures or not. It's handled now by XPress by explicit ligature encoding, which will tempt many developers to kludge a solution by coding ligature characters in the Private Use Zone. That's *not* what we want, is it?. It interferes with options like spell-checking, sorting, and searching. Transferability of text could be severely reduced as will be seen in the coding of *fi*, *fl*, *ffi*, and *ffl* in this PDF document in plain text, which uses the usual "second-font" ligature kludge. No harm is done by teaching the user that, to do ligation properly, a ZERO-WIDTH LIGATOR is used. Users have learned to employ SOFT HYPHEN this way already.

> There would be three kinds of cases (at least), and in all three of them, ligature rendering glyphs would be provided in the fonts.
>
> A. Use of ligatures controlled by global settings, as the above example for Latin "fi", as Devanagari, as Arabic.
> B. Blocking of ligatures in particular cases (using ZERO-WIDTH JOINER if cursive connections need to be preserved)
> C. Requiring ligatures in particular cases (for those, Old Hungarian and others which work similarly, triples like X + ZERO-WIDTH LIGATOR + Y would be built into fonts when the fonts support such ligatures).

> Everson is of course quite correct that the majority case for quite a number of alphabets is *non*-ligatured. Using Arabic or Devanagari or Latin “fi” examples simply *does not address* that situation, and is no counterexample to it nor any argument against it. Latin, Arabic, Devanagari need one sort of support, several alphabets named by Everson need another sort of support, and neither sort of support needs to exclude the other.

I don’t see how the VIRAMA/ZERO-WIDTH LIGATOR analogy doesn’t solve this. Look at Roman/Fraktur/Gaelic requirements (and indeed Turkic Latin requirements), which have quite diverse ligation requirements; but the support required is the same as that for Brahmic. I contend that Latin is no different from any other European script – the Gaelic, Fraktur, and Turkic examples should show this. Latin ligation cannot be automated.

> That is what I meant by saying there is no difference, that the use of ZERO-WIDTH LIGATOR adds *no new mechanism* to the functioning of the system, it only adds a single code which is to be disregarded on some level in processes like sorting and (optionally) searching.

I would say “(optionally) in processes like sorting and searching”. *Cf.* the discussion of bind-runes above.

¶

Peter Constable said:

> I wouldn’t necessarily expect each ligature to have its own (*i.e.* individual) style run. In general, it’s more likely that the font vendor has created the font to support varying levels of ligatures, and the user specifies a particular level of ligation for an entire paragraph or larger run of text, possible the entire document.

The range of options, as shown in Greek and Gaelic above, may be too complex for the designer to handle or predict. Better to leave the burden of ligature *setting* to the user, with automatic insertion of ZERO-WIDTH LIGATOR in trivial instances like Latin *f*-ligatures (with user deletion or special inputting settings for things like Turkic override of *f*- ligatures where necessary).

> In other words, the burden on the font designer is exactly the same, save for a minor detail. Thus it seems the implementations required to support ZERO-WIDTH LIGATOR vs. ZERO-WIDTH NON-LIGATOR should be fairly comparable. Some comments have suggested otherwise,

I don’t see a value in ZERO-WIDTH NON-LIGATOR. It requires a mix of levels and differs from the Brahmic model for glyph-cell selection. ZERO-WIDTH LIGATOR is like VIRAMA: if present it selects a glyph cell; if absent, no such selection is made. It is analogous the combining character model – if present, a precomposed glyph is selected; if such a glyph is not present, a base character and a zero-width combining character is selected (with possible bad centring of the diacritic).

> Marco Cimarosti said:
>
> > A non-historical example for the need to control ligatures in plain text has already been done: the “fi” ligature in Turkish. In most roman fonts, the dot over “i” disappears in the “fi” ligature, because it merges with the “f”’s top. This aesthetic adjustment is perfectly innocent in most languages, because the dot on “i” has no special meaning (it is just an heritage from handwriting). In Turkish, however, dotless “i” is a separate letter so, in certain fonts, the ligature loses the distinction between “fi” and “fı”.
>
> I’ve suggested in an earlier message in this thread that ideally all runs of text should be tagged to indicate their language.

Only a small subset of the world’s languages have been given taggable codes.

> If this is done, then it would be possible for that information to be used by the rendering engine in shaping the text and for the font developer to specify that the “fi” ligature *not* be formed for Turkish but that it be formed for other languages. (Current score, as I recall: OpenType already provides support for such language-specific substitution; such support in not currently available in AAT but is being considered.)

And how are the orthographic *rules* for all these languages to be accessed? Is there an orthographic register? (There is not, and even if there were, as shown for Gaelic above, it might interfere with the accurate representation of historical texts. With ZERO-WIDTH LIGATOR, you don't have this problem.) Up to 16 Turkic languages could make use of the Turkish *fi* convention discussed here. What happens when a language (like Greek or Irish) can be written in many fonts with different ligation requirements or options? Ligatures like *air* (air) will only be present in some Irish fonts, but no one would complain if switching from áoḋairıḋe in a Watts font led to a representation of aoḋaiꞃıḋe in a Newman font, since Newman fonts never had such ligatures traditionally anyway. Using ZERO-WIDTH LIGATOR to link character-sequences to glyph cells (ignoring the ZERO-WIDTH LIGATOR if no X-ZWL-Y triplet is available in the smart font) is by far and away the simplest solution to this chronic problem.

¶

Mark Davis said:

> A font vendor, right now, can map the unassigned characters from 2060-2069, E0000-E0FFF to the empty glyph. If we end up adding ZERO-WIDTH NON-LIGATOR and/or ZERO-WIDTH LIGATOR, then they will go in one of those positions. Rendering engines should (but may not) currently filter out all unassigned characters in this range, since their effect on rendering cannot be predicted. But once ZERO-WIDTH NON-LIGATOR or ZERO-WIDTH LIGATOR are approved and allocated, the rendering engine can pass them through to the font table execution code. Once that is done, then ZERO-WIDTH NON-LIGATOR would work with no further action by the font vendor. The vendor would have to add the extra ligature mappings to the font for ZERO-WIDTH LIGATOR to work. That's why ZERO-WIDTH NON-LIGATOR is simpler to implement than ZERO-WIDTH LIGATOR.

How does this work for eaᵹla/ęᵹla? What about the Runic example given above? There are *no* obligatory ligatures in Runic, so turning ligation on globally is obviously *not* a solution, and the model you and Asmus Freytag have suggested wtih ZERO-WIDTH NON-LIGATOR is very impractical for Runic, as it is for Latin-Gaelic or Latin-Fraktur. (Estonian Fraktur uses no ligatures. We could exhaustively investigate Fraktur as used in Danish, Swedish, Norwegian, Latvian, Lithuanian, Estonian, Finnish, and Sámi, to try to come up with a hierarchy of language-specific instantiations of rules, but the X-ZWL-Y model is far simpler for the font designer and the end user, and apart from possible input settings, is not dependent on further language-specific table implementations by programmers. Like SOFT HYPHEN, ZERO-WIDTH LIGATOR will support "languages that haven't even been invented yet". This glyph-selection model is already the one implemented for Brahmic conjuncts, Brahmic consonant + vowel-sign sequences, and Latin base-character + combining-character sequences.

¶

Kent Karlsson said:

> The typographic-aesthetic ligatures are *not* orthographically motivated [in Swedish], but typographically motivated: the (normal) glyphs overlap in an ugly manner, and if someone cares enough (especially for larger sizes), a ligature is formed that looks better. In modern use (or its lack of modern use at present for that matter) there is no orthographic implication whatsoever. Some maintain, apparently without evidence, that these ligatures do have an orthographic significance at morpheme boundaries. However, some example ligatures, given in some (Swedish) typography books, are for letter combinations that would, for Swedish, only occur for compound words: *ffj* and *ffb*, in incomplete example listings. In addition, the second g in, e.g., "tuggummi" (chewing gum), which subword does that second g belong to? Is it the last letter of "tugg" (chewing) or the first letter of "gummi" (gum)? Here these two g-s have been amalgamated into a single g. So what would be so horrible about doing a milder form of joining over a subword boundary? In the books on typography that I have looked in, they happily provide some examples of ligatures. There is no "by the way, don't use them over subword boundaries" or similar side remark.

Such rules *are* given for Fraktur German, as shown above.

> Instead it is taken as obvious that if you want to use ligatures to make the glyph sequence a bit better looking, you are free to do so anywhere in a word. They are, after all, only there for aesthetic reasons, and not ligating is no better looking over a subword boundary than anywhere else.

Then automatic insertion of ZERO-WIDTH LIGATOR in Swedish is always appropriate for certain letter pairs and triplets. No problem.

> 3. There has been no agreed upon way of accessing [such ligatures].

This is part of the problem with markup.

> So prior to the existence of AAT or OpenType fonts, it was hard and unsafe (for font changes) to use them, even if such non-English ligatures were present in a font.

That’s why we propose ZERO-WIDTH LIGATOR – to make it agreed and safe and portable.

> If a glyph sequence [...] has no overlapping glyphs, or the overlap looks ok anyway, no ligature should be used, and if a ligature is used anyway it should look exactly like the unligated glyph sequence. Any occurrences of preligated Latin characters from the FBxx block should be replaced by their compatibility decomposition, or otherwise look exactly like if it were compatibility decomposed (though the caret management will be different in the latter case). (An fi ligature should look different from an f followed by a dotless i. This is not an issue for Swedish per se, but may be for Turkish names, of persons or places, that are properly spelled in an otherwise Swedish text.)

So you delete ZERO-WIDTH LIGATOR for Turkish names. Simple! And expected for Turkish text.

¶

John Jenkins said:

> Michael would then be arguing that even though most writing styles for Latin don’t require explicit ligation control to be legible, some do. We need to add ZERO-WIDTH LIGATOR and ZERO-WIDTH NON-LIGATOR to support those writing styles.

No, we just need ZERO-WIDTH LIGATOR, and its absence. And this *should* be generalized for all typographic ligatures in Latin. Cost to the user? Minimal, since most current applications don’t deal with ligatures at all, and the most common *f*-ligatures could be handled with a simple all-on/Turkic/all-off switch (which *still* would insert (or refrain from inserting) ZERO-WIDTH LIGATOR in the requisite positions). Otherwise you maintain a mixed model for ligature production, which is untidy and unnecessary.

> Peter Constable said:
>
> > True, but I think the question still has some validity in the following way: In one scenario, a font designer has to list all sequences of the form “f” + “i” -> “fi” ligature “f” + “f” -> “ff” ligature while in the other scenario the font designer has to list all sequences of the form “f” + ZERO-WIDTH LIGATOR + “i” -> “fi” ligature “f” + ZERO-WIDTH LIGATOR + “f” -> “ff” ligature
>
> Actually, the type designer would probably want to provide both sets of sequences for those cases were ligation control wasn’t done by ZERO-WIDTH LIGATOR and ZERO-WIDTH NON-LIGATOR.

But it would be a lot simpler if it *were* done with ZERO-WIDTH LIGATOR for all languages in all European scripts.

> As an aside, on this I would absolutely insist – that ZERO-WIDTH LIGATOR and ZERO-WIDTH NON-LIGATOR be at most an *optional* way of specifying ligation control in plain text, allowing for ligation control in higher-level protocols without them. It would seriously undermine the technology that Apple’s been selling for five years to make them obligatory.

I really hate to criticize Apple, but who has bought and implemented this technology? Only Adobe? Even Apple doesn’t have a text editor available that implements it!

> > In other words, the burden on the font designer is exactly the same, save for a minor detail. Thus it seems the implementations required to support ZERO-WIDTH LIGATOR vs. ZERO-WIDTH NON-LIGATOR should be fairly comparable. Some comments have suggested otherwise, however; for instance, I seem to recall John Jenkins saying that implementing support for ZERO-WIDTH NON-LIGATOR in ATSUI would be fairly simple, but that he

> wasn't at all sure if or how support for ZERO-WIDTH LIGATOR could be added. Why should there be that much difference?

> The complication comes in trying to make sure that fonts *without* the ZERO-WIDTH LIGATOR version of the tables will work correctly. This will mean adding ZERO-WIDTH LIGATOR to the list of characters whose display we (almost always) actively suppress – things like the directionality overrides are rarely mapped explicitly to invisible glyphs in fonts, so we have to do this ourselves to make sure that black boxes don't litter the landscape of text that uses them.

True.

> The more difficult part is adding a step to parse the text for ZERO-WIDTH LIGATOR and, if it is present, checking the font to see if a particular feature allows for the equivalent ligature to be formed if there is no ZERO-WIDTH LIGATOR and overriding the user's collection of set features to include it.

True. That doesn't mean it's not the right thing to do. And it's not very different from what one has to do with base-character and combining-character sequences.

> This is similar to what we have to do in other situations. *E.g.*, if a font doesn't have a Unicode cmap, we make one and cache it, so that fonts will work "properly" even if they haven't explicitly been updated to support the new behavior. ZERO-WIDTH LIGATOR would require similar functionality on our part – old fonts should continue to do the right thing without being formally updated to do so.

They don't do this *now*, as I say – a major user of my Armenian Utiltiies has just said that he's starting to have problems with them on his new G4. I hope Apple can make tools available to help me update my fonts – but I think updating, ZERO-WIDTH LIGATOR or not, is *inevitable* given the tools we have at present.

> To work, you'd need to have:
>
> f + i --> fi
> f + SHY + i --> fi
> (f + ZERO-WIDTH LIGATOR + i) --> fi
> (f + SHY + ZERO-WIDTH LIGATOR + i) --> fi
> (f + ZERO-WIDTH LIGATOR + SHY + i) --> fi
>
> All explicitly in your list of potential ligatures in your font.

There's no harm in this, if it's really necessary. Font designers will welcome this, I should think.

¶

Peter Constable said:

> As we consider cost of implementations, I wonder just how many fonts there are in existence in each of these categories? *I.e.* what's the cost impact on vendors and users in relation to existing fonts of use of ZERO-WIDTH LIGATOR (or ZERO-WIDTH NON-LIGATOR) versus the cost of any other alternative? I suspect the costs in relation to existing fonts that Mark refers to is not very high (but feel free to correct me if I'm wrong) on the assumption that the technologies required to make smart-font ligation work (AAT and OT) haven't been around that long and/or have been utilised by only a very small number of applications.

I agree. Certainly even smart fonts and Apple's TEC remapping could break *all* currently available cmap-less non-Roman font kludges until Apple makes editable TEC software available to us. My WorldScript resources are already starting to break with System 8.6 on the G4, and at present I can do nothing about it to serve the users of my software.

¶

John Jenkins said:

> Both AAT and OT allow for bitmap fonts that have all the smarts needed to render anything in Unicode properly. The distinction isn't between stupid bitmap fonts and smart outline fonts, it's between stupid fonts and smart fonts.

Yes but there aren't very many smart fonts now (I can't make them), and as far as "anything in Unicode" is concerned, well, Unicode can't handle the requirements adequately without ZERO-WIDTH JOINER as far as I can see. You can represent what is actually in Unicode, but this doesn't support bind-runes according to the plain-text requirement, for example.

> In practice, GX and AAT don't filter out any unassigned characters before doing the character-glyph mapping, and I doubt that OT systems do, either. We *could* add them to the list of Unicode code points whose "empty box" display is actively suppressed, but that's a different matter – and we'd rather not, because that could potentially interfere with the last resort font and font substitution (or whatever we're calling it these days).

The Last Resort Font will have to be updated many times. Look at all the as-yet-unencoded scripts in the Roadmap.

> The *real* complication with ZERO-WIDTH LIGATOR is to update the rendering engine so that older fonts will continue to work "properly" without having to be updated. This is something we've been pushing really hard to do and we'd want to continue to do it. Realistically, however, we might be forced simply to punt and say that ZERO-WIDTH LIGATOR would only work with fonts explicitly updated to support its use.

If that's what's required, than that's what we should do.

¶

Lloyd Anderson said:

> I have always assumed that the technology which specifies a default (assuming the font supplies the glyphs), whether obligatory for Latin or not, would remain in place. (I assume this system does not choke if a font does not contain particular ligature glyphs, it simply renders without ligatures at that point...?) The use of ZERO-WIDTH LIGATOR and ZERO-WIDTH NON-LIGATOR, just like the use of ZERO-WIDTH JOINER and ZERO-WIDTH NON-JOINER (at least in most instances) would be special overrides, the second for overriding obligatory ligatures (which could be used in Latin, of course), [though I think ZERO-WIDTH JOINER would do for the ZERO-WIDTH NON-LIGATOR uses? if not, why not, perhaps I missed something earlier...]

I think I have dealt with this above.

> Tables of triples X + ZERO-WIDTH LIGATOR + Y would be needed *only* for [scripts like Old Hungarian Runes, Germanic Runes, etc.], they would not be needed for Latin, and would not appear in Latin fonts? At least not for ligatures which were included in the obligatory set.

I disagree. What is the "obligatory" set? What about Turkic?

> I hesitate to speculate (given my limited knowledge) further concerning any, to me still hypothetical, need for ZERO-WIDTH LIGATOR for Latin. Though I can conceive that there might be a split between normally-obligatory ones and optional ones, that still could be handled as for Arabic script, by having global settings to include none/only obligatory/obligatory + preferred/all (or whatever the details of the current implementation are).

There are too many potential options to make this an attractive solution. ZERO-WIDTH LIGATOR is simpler.

¶

John Fiscella said:

> A method of ligature control has been suggested using ZERO-WIDTH LIGATOR and which would require a smart font to intervene in order to exercise complete ligature control. While I am not opposed to the introduction of the ZERO-WIDTH LIGATOR (even enthusiastic that they would be available), as a font architect and font designer, I feel that we mixing apples and oranges if we require the ZERO-WIDTH LIGATOR to only be usable through a smart font mechanism; because it is inconsistent with present complex script rendering conventions.