ISO/IEC JTC1/SC2/WG2 N2141
1999-12-01

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

Doc Type:	**Working Group Document**
Title:	**On the need for a ZERO-WIDTH LIGATOR**
Source:	**Michael Everson**
Status:	**Expert Contribution**
Date:	**1999-12-01**

Recently Mark Davis took up again (on the UNICORE@UNICODE.ORG discussion list) the question which had been discussed and dropped a little while back. Since I appear to have invented the unfortunate term GLUER, I hereby withdraw it, and have globally replaced it with LIGATOR in all the texts cited below, as it is more accurate. Mark cited Rick McGowan:

> I always start with the assumption that ligatures, except in *extremely* rare cases that are dictated by the writing system itself, are optional.

I would beg to differ. 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.). Those are the most common ones; the other *f*-ligatures, *e.g. ﬃ* and *ﬄ* ligatures, etc. are also found in good Latin typography and should be applied where appropriate. One may question the notion of *good typography*. To that I say: it is only relatively recently that technology like typewriters and dot matrix printers and 8-bit glyph limited fonts have forced *bad typography* on us, and it is only thence that the notion that the *f*-ligatures are optional has arisen.

> The default behavior in many environments should be to not use ligatures. They should be applied as a typographical device used to make text look nice or flow in a particular way during a layout and page design process, not during the drafting of the text itself, and so forth.

> Rick, you may have that assumption, but it simply doesn't match reality. People want to turn on normal ligatures such as *ﬁ* globally, then correct the very few instances (if any) that don't work properly. They don't want to have to manually scan through every possible place that ligatures could occur and do some special action.

In general, this is true. But to effect such an option, the text has to be coded somehow at input, whether by hypertext markup, or (as advocated here) by plain text encoding.

> This is the same as automatic hyphenation. You want to turn on hyphenation globally, then correct the few instances where it doesn't work manually.

I never do this. I always prefer to avoid hyphenating words and to manually apply hyphens only when the white space looks too big to my eye. I don't trust the algorithms, as they have no eye for what is best.

> For that matter, this is also true for BIDI; the implicit algorithm should work fine the vast majority of the time, but you need manual controls to fix the few cases that don't.

True. But this is a different question from *how* to store the information in a document. We need a LIGATOR that can easily be used to specify to Latin ligatures and other European script ligatures alike. But here is where it begins to get contentious. Lloyd Anderson wrote in response to Mark some things with which I disagree mightily, although Lloyd and I seem to agree in principle on the utility of a ZERO-WIDTH LIGATOR. He said:

> Ligatures are indeed automatic in many contexts of interest to us, that is to say, writers/readers want them to be automatic, to not have to specify them individually. Only the abnormal cases call for manual overrides.

But the manual overrrides may be “Oh, don’t ligate here because it crosses a word boundary” (the hypothetical *_chafffinch_ must be typeset _chafffinch_ [cha(ff)(fi)nch] not _chafffinch_ [cha(f)(ffi)nch]), or “Please ligate this Runic pair even though it’s very rare.” The point is that the encoding should carry this information, however it is input and/or deleted.

> Perhaps I have misunderstood, but I think a global setting has always been Rick’s position, and he took it so for granted that he was only expressing opinions on how to handle the abnormal cases.

Either way, a reliable and predictable and *interchangeable* means for specifying ligatures is required. Arabic always ligates unless the ZERO-WIDTH JOINER or ZERO-WIDTH NON-JOINER intervenes. Devanagari always ligates when VIRAMA causes the ligation, unless the ZERO-WIDTH JOINER or ZERO-WIDTH NON-JOINER intervenes. But for European scripts we have nothing specified but that either the font or the application should handle it… *somehow.*

“But lo!” sayeth the user, “there is block FB. We can use what’s there! Except look, it’s missing dozens of ligatures, so we’ll have to add all of them, now where’s that list…”

One cannot have it both ways. Either we have a stable and productive and *interchange-able* method for ligating Latin and other European scripts, or requests for additional ligature characters in the UCS will continue to be put forward. A safe and reliable mechanism for specifying ligatures is a requirement both of the font designer and of the end user. The text is no more burdened by a ZERO-WIDTH LIGATOR than it is by a SOFT HYPHEN, and the relation of character-sequence to font glyph is analogous to that which we already have with combining character sequences.

> (I think Rick’s position was questioned by Asmus pointing out that we want plaintext to handle this kind of thing, so do need a local in-text method of handling unpredictable cases, *other* than applying a scoped formatting. Such a critique presupposes that the general case is a global setting, whether formatting (Rick?) or built into the font defaults.)

This presupposition is part of the problem, and anyway it appears to me to be input-related, not encoding-related. The problem is that the presupposed general case doesn’t work for all European scripts.

> But there are also other contexts where there is no plausible automatic rule, where the choice is indeed completely local. These are the ones which have been highlighted by Everson, from Runes and similar cases. Neither kind of global setting works here with any plausibility, that is, neither a global formatting default nor a default built into the fonts. We need to specify the ligatures as if we were specifying separate characters in *these* cases (for which experience with

> the automaticity of *fi* or Arabic or Devanagari are not directly relevant), simply because the occurrences *are* entirely local, not predictable in any automatic way.

The encoding for ligatures of *any* kind, and for *any* script, whether Latin, Runic, or Greek (or other particular scripts to be specified), should use a ZERO-WIDTH LIGATOR in every case, whether certain inputting settings insert it automatically in some of those cases or not. And Lloyd is wrong; *fl* may be universally automatic in Latin, but *fi* is not, as in the case of Turkish and Azerbaijani. It is not the application of any particular ligature which is the question. All ligatures in Devanagari are specified in the same way: by use of the VIRAMA.

> So we need a LIGATOR to do that *in these cases*, its use should be deprecated in those cases where users want the ligatures to be handled automatically.

No! Ligation is a generalizable behaviour. Treating all ligatures the same way is the only logical and *interchangeable* thing to do. An *ꝏ* ligature is no different from an *fi* ligature, even though the latter is far more common.

> It is perhaps our use of the term “ligature” for both kinds of cases, undifferentiated, which leads some to think we should handle all of them the same way.

Of course we should handle all of the same way, just as all Devanatari ligatures are handled in the same way. Ligature *availability* is a matter for the font (a font either has the glyph or it doesn’t). There is no *fi* ligature in the Gaelic variant of the Latin script: for the sequence ꝼı, the *i* has no dot and the *f* does not overhang it. A text which encodes the sequence as F + LIGATOR + I but which has no font ligature will just display *fi*. Ligature *presentation* however, should be specified in plain text by ZERO-WIDTH LIGATOR. Otherwise they should (for European scripts) remain latent.

> They are different phenomena, so we should not be surprised if different mechanisms are appropriate. A global setting will not work at all for the Runic ligatures case. Rick’s formatting of strings of text will work, but is rather laborious, and as I understood Asmus to be pointing out, is not plaintext and we want to be able to represent normal Runic text (just for one example) in plaintext.

And to represent Latin, and Cyrillic, if, for example, *рх* (used in Moksha earlier this century) is to be considered a ligature of *p* and *x* and not a character as in ISO 10754. (NOTE: Printed Cyrillic almost *never* makes use of ligatures if the font style isn’t Old Church Slavonic. I do not believe that *рх* is a ligature, but rather a letter like *æ* (used in Ossetian). ISO 10754 encoded *рх* and *лх* as characters, and these should in my view be encoded as characters in the UCS. In any case there are plenty of ligatures in OCS printing.]

> So we need a LIGATOR to do that. The non-ligatured forms are the default in Runic, I believe, and the ligatures need to be individually specified when required, by entering a LIGATOR code into the text stream at just those places, and by having triples of A + LIGATOR + C in the fonts for any such ligatures supported by the fonts.

Such triplets are necessary for all such ligations. Also quadruplets can be taken into account: LATIN SMALL LETTER F + LIGATOR + LATIN SMALL LETTER I + COMBINING GRAVE ACCENT must yield *fì* not *ﬁ̀*. Costs? Technologies like ATSUI (as yet unimplemented for most fonts) may have to be altered slightly to use ligature triplets, and globally turning ligatures on or off would be some sort of global search and replace operation. This may differ from how

ATSUI deals with it now, though ATSUI *does* interact with character strings (such as base and combining characters), and so it is probably not very costly even if it does imply a change. The point is even if ATSUI can turn ligatures on or off at one or more levels, this doesn't work in a generalizable way.

§ § §

All of the above was a response to an earlier summary and discussion which I submitted on 1999-11-07. I reproduce it here, with a few augmentations. Lloyd said:

> As I understand it, at least many years ago, the opposition to a "LIGATOR" was present at the same time that Joe Becker's very sophisticated and precise definition of the functioning of the ZERO-WIDTH JOINER and ZERO-WIDTH NON-JOINER were being presented (which I happily admit I did not understand at first, but strongly supported once I did).

Chapter 13 of Unicode still proscribes the use of ZERO-WIDTH JOINER for scripts like Latin.

> Becker's point was that both of these interrupted the flow of other codes, causing codes which would have been adjacent to be not-adjacent. The ZERO-WIDTH JOINER also would separate adjacent characters, but it would have the special cursively linking character property, so characters adjacent to it would take their cursively linking forms on the sides adjacent to it, if they had any.

For scripts like Arabic, Syriac, or Mongolian, which are inherently cursive, it makes very good sense. So its proscription for Latin is probably quite sensible – except for the peculiar references in the Unicode Standard to "cursive fonts"; see below.

> Now the LIGATOR is different. While it *also* separates any characters which it occurs between (obviously), it is desired that inserting this character should cause the two adjacent characters to form a ligature together.

Such a ligature is displayed if the ligature is available in the font. Note that we also have three script-specific modifiers for Mongolian which cause specific glyph selection. The ZERO-WIDTH LIGATOR should also be script-specific – though it may apply to more than one script in the UCS (not Katakana or Egyptian Hieroglyphs, for instance) it should not be applied to all of them.

> If the LIGATOR is to be simply another text character, then for it to work, smart fonts must include listed triples of characters A + LIGATOR + C and the single glyphs which are to be substituted for such triples.

That's right. And quadruplets as noted above. Rick McGowan wrote:

> NeXTStep implemented this concept several years ago; it was handled by the glyph-producing code for specific fonts under Display PostScript. It works just fine. The text editor has 3 menu items which can be applied to the current selection. The ligature level is stored as a numeric attribute over a run of text. The default choices were "no-ligatures" (*i.e.*, only absolutely required ligatures like LAM-ALEF), "standard ligatures" (like *fi*/*fl*) and "all possible ligatures".

Three menu items is *not by any means* a sufficient number of categories for specialized needs, though it is OK for some fun calligraphic fonts (usually used as examples of how the Display PostScript (etc.) technology works). But typographical ligatures like *fi*/*fl* are part of the *ordinary* behaviour of fonts like *Times*. And what happens to *interchangeability* if *e.g.* ATSUI (the QuickTime GX analogue or descendent of the NeXTStep implementation, I suppose) is not implemented in a given environment?

> Since it was specifiable over a range, you could use it to do optionally fancy ligatures in certain places within a document. It could also be programmatically manipulated by applications for specialized typographical purposes and so forth.

This is impractical for nonce-ligatures, and all the work the user has done to format such runs of text is lost when the document is converted to plain text.

> There was no need for a LIGATOR character in this scheme. This is a good example of using “rich” or “attributed” text to get a reasonable effect that’s not appropriate for plain text.

Why is the use of encoded ligation appropriate for some scripts and not for others? The point is that the “reasonable effect” Rick refers to is not comprehensive enough and that plain text representation is preferable. In Arabic, Devanagari, and Mongolian, ligatures are treated in explicit if differing ways. For Latin and Runic and the rest we have no mechanism. Why? There are even nonce ligatures in Scottish Ogham inscriptions – but no means of representing them in UCS encoding.

John Fiscella wrote:

> I personally would not like ligature selection/management mechanisms built into “smart fonts.”

I am not quite sure what exactly John means here by “built into”, but the discussion below may clarify things. I suspect that John agrees with my analysis and that he would like the *encoding* to specify ligature behaviour.

> All the ramifications of the mechanism may not be aware to the originator at the time of creation. (The old slogan: “Computers are dumber than people but smarter than programmers.”) Can an end-user reprogram the ligature management mechanism in a font if it turns out to have an unintended nasty consequence?

The end-user probably cannot. 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.

I know that Apple’s ATSUI has some sexy features (like you could build into your Chancery font a way of *preventing* the user from using it in ALL CAPS). But many of those kinds of typographical features are just for fun. Ligation in Latin script is a serious concern of mine, especially because of my work on early Gaelic typefaces, which make a *much* greater use of manuscript ligatures than nearly any Roman typeface did. Early Greek typefaces did likewise.

I consider typographic Latin ligatures to be quite analogous to combining characters. In the font, I need only design a single LATIN SMALL LETTER E, for instance, and I can use internal linking tables to paste its glyph into glyph cells which it shares with correctly-placed accent marks (also pasted in from elsewhere in the font, usually linking them to strings of UCS characters). I can do this even now, with Fontographer. At present, because my Unicode capability remains limited, I haven't experimented with linking glyph cells to strings of UCS characters (*i.e.* ḛ LATIN SMALL LETTER E + COMBINING TILDE BELOW) but I know that there are font tables that can do this.

These tables are really nice, because since combining characters are highly productive, it means that while most fonts may only handle a standard set of (Western and Eastern European) vowels and consonants combined with accents into dedicated glyph cells, designers like John and Lloyd and I could make way-cool fonts that could handle all the letters used in indigenous American languages, African languages, phonetic fonts with all sorts of tones and aspirations, and so on. The character string LATIN SMALL LETTER E + COMBINING TILDE + COMBINING ACUTE ACCENT + COMBINING TILDE BELOW (nasalized high-tone creaky-voice e) mightn't look well in a bog-standard *Times* font which doesn't have such special glyphs, but the same could look quite nice in *Utopia Amerind Italic*: *ḛ̃́*.

This shows a way we have of relating font technology with character strings. It works great, or will do so, when the technology catches up. And this way of specifying glyph relations to strings of UCS characters works in plain text.

Now let's think about ligatures in Latin. Mac Roman fonts have had *ﬁ*/*ﬂ* ligatures from the beginning. Some Mac applications know about them. Quark XPress, for instance, lets you turn them on or off. I always have them turned off because I also use Mac Gaelic and Mac Icelandic fonts quite regularly and Quark, since it thinks all non-WorldScript fonts are Mac Roman, performs unwanted substitutions (*ŷ* or *þ* for *ﬁ*) if the feature is enabled. (And no, the end-user can't fix it.) I have not tried to determine how Quark represents this in coded text, but I assume it is done by markup as Quark's italic is. (That would be what Rick suggests anyway.) But style markup can be misleading. For instance, if you use the ALL CAPS style, your text may display in all caps, but still be encoded as lower-case characters. So marked-up plain text may be different from WYSYWIG even in a context one wouldn't expect.

Asmus Freytag wrote:

> The kind of scheme put forth by Rick fails badly for many languages, where the availability of ligatures is not determined by the character codes in context, but also by the meaning expressed by these characters. Ligatures are typically disallowed across certain morpheme boundaries in the middle of composite words.

That's right. It also fails for languages where ligatures are rare or uncommon and no rules whatsoever apply. One could deal with German or Estonian or Danish Fraktur ligatures algorithmically with dictionary lookup if necessary. One cannot do so with Runic or Old Hungarian, because standard orthography does not exist for the language corpora they are used for and it would be pointless to try to create such dictionary lookup tools.

> The absence of a cheap clean override for automatic ligation is the single biggest reason for the pressure behind the Latin ligatures in Unicode.

Yes. Asmus is right. Rick McGowan disagreed:

> No, you misunderstand completely. First, the availability of ligatures is determined by the designer of the font and the implementor of the particular font file.

Yes, but the repertoire of possible, acceptable, or nonce ligatures is external to the font designer's implementation in many cases. He or she needs to get his or her tables from somewhere else; such repertoires of ligatures have an independent existence in most cases (the Runic corpus, for instance).

> The *usage* of ligatures is determined and modified by the creator of the text, obviously with some particular [linguistic] context in mind.
>
> *E.g.*, in using this system for a German-speaking environment you would just turn off ligatures normally by default, so that you don't get the "fff" problem and so forth; and then if you *want* a ligature some place, select the text and turn on ligatures, at an appropriate level, for that bit of text.

The problem is that an unnecessary burden is placed on the user in such an instance. Not all languages use ligatures in the same ways at the same time or in the same measure. We've been focussing on the Latin script typography, but the use of typographic ligatures in Greek, Cyrillic, and Armenian is just as dependent on period.

We are all familiar with the huge number of ligatures used in handwritten manuscripts, and in general these are expanded in italics by editors. And in general, Roman Latin typography does not make use of an enormous number of ligatures. But Gaelic Latin typography and Greek typography did employ a very large number of ligatures in metal type from the time of the very first printed works. As time went on, use of ligatures was reduced, but *not* systematically. Some fonts had many ligatures, some had few. It is not at all practical to use a markup system to try to deal with different levels of ligature behaviour because there are too many of these for users or font designers to learn. A three-fold or five-fold hierarchy won't give the right behaviours; I have two bibles printed in Watts' 1818 Gaelic font (revived by myself as *Acaill* and used for Gaelic examples in this document), one which uses all the ligatures, one that uses only some of them. Other texts using the font employ no ligatures at all.

What I want to see as a font designer *and* end-user is a character very much like the SOFT HYPHEN. The SOFT HYPHEN character can appear peppered throughout a text and is ignored unless a break occurs at the end of a line. If we had a LIGATOR, this could also be ignored unless the font had glyphs (as Lloyd described) which were composed of the sequence *e.g.* F + LIGATOR + L.

This isn't just for representation of old printed texts. Ligatures like *fi*/*fl* are common; but *fh* ligatures are needed in modern Irish (Roman face) typography because *fh* is a very common string in Irish (it represents phonetic ∅; cf. Latin *fabulare* > Spanish *hablar*). I don't know of a single font that has this ligature, probably because the string *fh* is uncommon in most Latin-script languages. If *fi*/*fl* ligatures were coded as F + LIGATOR + I and F + LIGATOR + L then the font will display them if the ligature exists in the font's glyph tables (linked to the character triplet sequence), and simply display as F + I and F + L if not. If F + LIGATOR + H occurs, the same thing will happen. Inputting with ligating turned on could insert the LIGATOR algorithmically, perhaps according to a user-editable table (*cf. fl* but *fi* and *fi* in Turkish and Azerbaijani). But even if an application didn't know about *fh*, then the user could insert the LIGATOR manually or globally after writing his or her text. Not dissimilar to SOFT HYPHEN implementations currently available.

The point is that coding the LIGATOR would be more advantageous than unstandardized markup, because the ligatures would be specified in plain text. It would do no harm for fonts without ligatures, but would be noticed by any font with glyph tables that contained the triplets (just as they do for accented vowels).

Common ligatures in Gaelic fonts: *air*, *ar*, *ea*, *ñ* (= *nn*), *rr*, *ui*, *úi*.
Less common ligatures in Gaelic fonts: *gur*, *li*, *ll*, *nea*, *s̃* (= *ċt*), *sd*, *sg*, *si*, *sn*, *so*, *st*, *tt* (and there are many more).

Lee Collins cited me with a comment:

> In Runic and Old Hungarian they are not "typographical devices", they are historical nonce-forms or more-than-nonce-forms which are nonetheless non-obligatory (in the way that ligatures are in Devanagari and Arabic).

> Whoa! KSA, TRA, and JNA are all obligatory ligatures in Devanagari. They even appear in the var.n.amula (alphabet charts). But both ISCII and Unicode happily and successfully treat them as ligatures. How are the Runic and Old Hungarian cases different?

I meant "in the way that ligatures are obligatory in Devanagari and Arabic". ISCII and Unicode rightly treat all Devanagari conjuncts as ligatures. They make use of VIRAMA to produce them (by using the triplet TA + VIRAMA + RA to yield TRA in the glyph tables) coded in plain text. In Arabic, ligatures just happen (also by reference to strings in glyph tables) but they can be prevented with ZERO-WIDTH NON-JOINER and forced with ZERO-WIDTH JOINER. In Latin, we have no such mechanism. Unicode 3.0 chapter 13 states for Arabic: "By providing a non-joining neighbor character [irrelevant for Latin] where otherwise the neighbor would be joining, or vice-versa [relevant for Latin], they deceive the rendering process into selecting a different joining glyph."

Also: "[A]ny number of ZERO-WIDTH NON-JOINER or ZERO-WIDTH JOINER characters sprinkled into an English text string will have no effect on its appearance when rendered in a typical non-cursive Latin font." Does this mean that these characters will effect a cursive Latin font? How is a Latin font supposed to announce to the application or rendering engine that it is "cursive"? What if one has a non-cursive Arabic font? (There were experiments.) How is such a font to be identified?

Also: "Usage of optional ligatures such as *fi* is not currently controlled by any codes within the Unicode standard but is determined by protocols or resources external to the text sequence…."

Why this restriction is imposed on Latin (Greek, Armenian, Cyrillic, etc.) non-cursive fonts? Typographical ligatures are of primary interest to *non-cursive* Latin fonts, and *cursive* Latin fonts are relatively rarely used.

The situation for Runic and Old Hungarian is that the normal representation is without ligatures, but (many) documents occur in which two or more of the letters are ligated. For a Runic text containing 50 ᚢᚾ URUZ + NAUDIZ sequences, two of them might be ligated.

Ligatures in such texts occur for reasons of saving space in the inscription, or at the afterthought (*i.e.* correction) or whim of the scribe. There is no pattern that can be used to define algorithmically such ligation since it is nonce or semi-nonce, and is not restricted to single texts.

Since ZERO-WIDTH JOINER and ZERO-WIDTH NON-JOINER are proscribed for forcing Latin, Runic, or Old Hungarian ligatures. Since these cannot be used, then we need ZERO-WIDTH LIGATOR, because ligature behaviour in European scripts is as important as it is in Brahmic or Arabic scripts (or at least is a valid user-requirement). We've just added three glyph modifiers for Mongolian. We need only one for European scripts.

Ligation should not be considered a stylistic feature in European scripts if it is considered a plain-text feature in other scripts.

Asmus Freytag wrote:

> In addition, intra-word markup is really a poor design. Experience has shown that it regularly interferes with non-layout text processing, whereas single character override functions such as SOFT HYPHEN, ZERO-WIDTH JOINER, ZERO-WIDTH NON-JOINER and ZERO-WIDTH NON-BREAK SPACE are a well understood concept and all algorithms must be set up to handle them already. If we are now in the middle of adding ZERO-WIDTH WORD JOINER or ZERO-WIDTH WORD SEPARATOR it would be the right time to add the ligature related characters at the same time, to allow us all to get on with life and give similar level concepts the same level of expression and a similar processing model.

This will make implementation lots easier for the font developer and for the end-user and even for the programmer. It seems to me that we need to add the ZERO-WIDTH LIGATOR, and to prepare a UTR describing its nature of the and function.

Ligation is a normal if sometimes unpredictable feature in the following European scripts: Armenian, Cyrillic, Greek, Ogham, Old Church Slavonic, Old Hungarian, Runic. Advantage of encoding ZERO-WIDTH LIGATOR: easy support for ligation in all these scripts. Disadvantage: Continued confusion for all these scripts and reliance on non-transferable incompatible markup.