ISO/IEC JTC1/SC2/WG2 N2317
2001-01-19

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 use of JOINERs in ligation**
Source: **Michael Everson**
Status: **Expert Contribution**
Date: **2001-01-19**
Distribution: **WG2 and UTC**

The question of character-based representation of ligation has once again been raised. In N2141 and N2147 I showed evidence for why a non-plain-text mechanism was required to force ligation of two adjacent glyphs. I proposed the addition of a ZERO WIDTH LIGATOR (ZWL) to force, for example, ligature ɑıꞃ *air* by coding ɑ + ZWL + ı + ZWL + ꞃ for Irish Gaelic. At the end of N2147, I wondered if ZWL could be used to force Arabic ligation which went beyond the ordinary cursive connections required for Arabic. (I have changed the examples given in the original document for clarity.)

> **14. Ligatures in Arabic.** I am not arguing strenuously on this point, but one possible additional use of ZERO WIDTH LIGATOR could be to assist in Arabic ligature selection. This is *not* the same thing as Arabic shaping behaviour. So, given ص + ح + ح = صحح, it might be possible to use ZERO WIDTH LIGATOR to select true ligatures, so: ص + ZWL + ح + ZWL + ح = صحح; ص + ZWL + ح + ح = صحح. The Arabic model may not require this, however.

The UTC decided not to add a new ZWL on foot of arguments by Mark Davis which indicated that ZERO WIDTH JOINER (ZWJ) could be used to do this job. The arguments, presented in L2/00-025, are reproduced here; again I have altered Mark's examples slightly, and I have added italics here and there.

> **1.** The following is a proposal to satisfy the desire of some to have more control over ligation with Unicode. It does this by extending the definition of ZERO WIDTH JOINER (ZWJ) and ZERO WIDTH NON-JOINER (ZWNJ). *Currently*, the description of ZWJ and ZWNJ specifically excludes ligation. That is, f + ZWJ + i is not to create a ligature, but only to join the characters cursively (if the font allows). Thus ص + ZWJ + ح will cursively connect صح, and the ZWJ would not have any effect on ligation. We are somewhat inconsistent with this in the Indic world, since conjuncts are really more akin to ligatures than they are to cursive connection.

I don't agree with this last point; Indic scripts make use of the VIRAMA in addition to ZWJ and ZWNJ, which is a means they operate under a different shaping model. Nevertheless, I've never liked this very much. Most, or a great many, cursive Latin fonts with joinings are "dumb fonts" designed so linking strokes meet neatly, just as in lead typography. (Remember the IBM Selectric script font golfball?) The feature of cursivity as applied to scripts other than the Arabic-derived ones should be investigated. But I will not do so in this document.

There have been experiments with non-cursive Arabic forms (Roberto Hamm, *Pour une typographie arabe*, Paris: Sindbad). Cursivity should be font-based, always for Arabic, rarely for Latin or Cyrillic. I am not convinced that ZWJ should be recommended for Latin or Cyrillic cursive fonts.

2. The reason we kept them separate was for Arabic, where one could want cursive connection without a ligature. However, a broadening can be done without materially affecting the usage in Arabic. Here are the definitions I propose.

3. ZWJ – if possible, produce a more connected rendering of adjacent characters than would otherwise be the case. In particular:

3.1. If two characters could form a ligature, but do not normally, ZWJ requests that the ligature be used. (If there is no ligature [in the font], and the characters normally would normally cursively connect, ZWJ has no effect.)

f + ZWJ + i => fi

ص + ZWJ + ح => صح

ص + ZWJ + ح => صح (if there is no ligature in the font)

3. 2. If two characters could cursively connect, but do not normally, ZWJ requests that they cursively connect. In particular, if a character X on [one] side has a cursive form, and the other character Y does not, ZWJ requests that X take a cursive form.

ص + ZWJ + [space] => [space]صـ

4. ZWNJ – if possible, break both cursive connections and ligatures.

f + ZWNJ + i => fi

ل + ZWNJ + ا => لا (where ل + ا normally = لا)

ص + ZWNJ + ح => صح (where ص + ح normally = صح)

5. In other words, given three broad categories:

2: ligated
1: cursively connected
0: unconnected

ZWJ requests that glyphs in the highest available category be used; ZWNJ requests that glyphs in the lowest available category be used.

6. For those unusual circumstances where someone wants to forbid ligatures in a sequence XY, but promote cursive connection, the sequence X + ZWJ + ZWNJ + ZWJ + Y will work. The ZWNJ breaks ligatures, while the two adjacent joiners cause the X and Y to take adjacent cursive forms where they exist. Similarly, if someone wanted to have X take a cursive form but Y be isolated, then the sequence X + ZWJ + ZWNJ + Y could be used (as currently).

ل + ZWJ + ZWNJ + ZWJ + ا => لا (where ل + ا normally = لا)

ص + ZWJ + ZWNJ + ZWJ + ح = صح (same as ص + ح)

ص + ZWJ + ZWNJ + ح = صح

7. Implementation
For modern font technologies, such as OpenType or AAT, font vendors should add ZWJ as appropriate. ZWNJ will probably have the desired effect naturally, without any change, since it is doubtful that non-Arabic fonts deal with it at all, except to map it to an invisible glyph.

"Probably" might not be precise enough. We should *know*.

8. Current Arabic shaping algorithms should work as they are; optional ligatures just would not be promoted by ZWJ, but current text should not be affected. The reason is that the current use of ZWJ between characters that normally cursively connect is redundant right now (as a matter of fact, with bad implementations of ZWJ or unsupported ZWJ, the cursive connection would actually be broken), and should occur in very few instances. If this proposal is accepted, then the additional semantic of promoting ligatures can be added to Arabic implementations over time. The worst that would happen is that a current redundant usage of ZWJ would cause an optional ligature to form.

9. Pros
- The characters exist right now, and could be applied immediately, without waiting for the addition of a new character.
- The current use in Arabic would not be substantially affected.
- The number of format characters, which require special implementation handling, is not increased.
- The behavior is more consistent across all scripts, including Indic.

I don't see how Indic is affected. A Devanagari font may have two available glyph shapes for क्क and क्क. But use of ZWJ with VIRAMA is specified in the Unicode Standard.

क + ् + क = क्क
क + ् + ZWJ + क = क्क
क + ् + ZWNJ + क = क्क

This isn't universally applicable to Indic scripts, by the way. Only a very few of them make conjuncts with easily-parsible "half-forms". Compare the same sequences in Myanmar, which has no half forms. What does ZWJ do?

က + ် + က = ကျ
က + ် + ZWJ + က = ကျ or ကျက?
က + ် + ZWNJ + က = ကျက

Recently, Tom Milo asked me the following, which indeed was the impetus for writing this document.

A question related to the COMBINING GRAPHEME JOINER (CGJ). We are designing a high-end typographic Arabic shaping engine for conventional (static, OTF style) fonts. By default it is going to churn out ligatures wherever possible. In order to break them down to "typewriter" style Arabic, we need an input code that interrupts the ligature selection: a COMBINING GRAPHEME DISJOINER. Now the question: has this GRAPHEME DISJOINER been considered before and if so what was the verdict?

I'll return to this below, with a possible answer. Before that, though, let's look at the CGJ. Peter Constable said:

> Some comments in relation to two earlier messages. Mark Davis wrote:
>
> > The two purposes of the CGJ are:
> >
> > 1. bind characters into a grapheme for the purpose of grapheme counts and collation. This would allow, in plain text, the same data to have "ch" considered as two graphemes, and "ch" considered as a single grapheme (with an invisible CGJ between them).

A good example of this is Welsh, where the grapheme <ng>, when pronounced [ŋ] as in *englyn* ['ɛŋlɨn] 'Welsh alliterative stanza', sorts as a single letter of the alphabet between <g> and <h>, but when pronounced [ŋg] as in *engrafu* [ɛŋ'gravɨ] 'engrave' is treated just as two ordinary letters. The following shows the alphabetical order:

> *egwyl, englyn, ehedfa, enfawr, engrafu, enhuddo*
>
> 2. as a side effect of #1, since combining marks apply to graphemes as a whole, it can be used to apply a combining mark to a sequence of letters that form a single grapheme.
>
> a + CGJ + i + ◌͡◌ => i͡a
>
> Both #1 and #2 are independent of whether or not the characters form ligatures... So CGJ and ZWJ are independent...

True, but they may interact, and this interaction is one reason Ireland objected to the introduction of the CGJ until such time as the ramifications of that interaction have been sorted out.

Consider Fraktur. Standard Fraktur fonts include 𝔠 *c*, 𝔥 *h*, and a ligature 𝔠𝔥 *ch*. But in Czech the two letters may occur at word boundaries. For example *viachlasný* 'polyphonic', which is compounded of two words (*viac* 'more, many' and *hlasný* 'vocal') has a genitive plural *viachlasných* 'of (more) polyphonic'. In Fraktur this is written viachlasných, where the first pair has no ligature because there is a word boundary and the second is ligated. But it may also be COMBINING GRAPHEME JOINERed. So how do the ZWJ and the CGJ interact? What must the font designer, spell-checker, etc. do?

> Ken Whistler wrote:
>
> > What Michael and I discovered is that there is a third important consideration. As a side effect of #1, a CGJ could result in making a sequence of combining marks be considered a single grapheme, for the purpose of application to a single letter. This could eliminate the need for encoding any side-by-side diacritic combinations...
>
> First, I don't know that we want to talk about sequences of combining marks as a single grapheme, since we have generally come to use that to refer to abstract character sequences that are conventionally recognised as a distinct element of an orthography by users and that function as a distinct element (e.g. for purposes of "character" count and collation). Sequences of combining marks are not necessarily (and not usually) graphemes in this sense. This makes me wonder if "GRAPHEME " JOINER is the right name.

What we have here with diacritics is several cases. There is a need to override the default diacritic stacking, for instance.

In the Finno-Ugric Phonetic Alphabet (FUPA), diacritics may stack or they may stand side-by side. In the IPA, diacritics may link together when combined for tone contours. The types of combination are given here:

s + ◌̌ + ◌̀ = s̏
s + ◌̌ + CGJ + ◌̀ = s̏̀
s + ◌̌ + ZWJ + ◌̀ = ŝ̀

In Greek, we have a similar situation, since by default Greek diacritics sometimes stack and sometimes do not stack, depending on the case of the letter or what letter it is. But what does a font do when it doesn't have precomposed glyphs but where the polytonic characters are coded as combining characters? Perhaps the rule here is Greek fonts have to have precomposed glyphs to work properly – just as Arabic does.

Still, Peter's query, Does CGJ have an orthographic semantic which is different from the visual behaviour here? I am not sure whether ˇ` is to be considered a grapheme in FUPA, but I wonder if Vietnamese ấ provides a clue. Here â is the base letter to which tone marks are applied. We shouldn't expect that Vietnamese texts be burdened with CGJs everywhere, as applied to FUPA. (It doesn't yield the right glyph positioning in any case.) On the other hand, we do have the Vietnamese special accents encoded – but are these not deprecated?

Peter continued:

> Secondly, it occurred to me that we're conflating two categories of control: Mark has described in #1 a control that affects determining what is a unit in relation to orthographic functioning. In contrast, what Ken and Michael describe is a *visual* control (though still different in effect from the effects of ZERO WIDTH (NON-)JOINER). Mark's #2 is a combination of both. I just wondered if we're sure that we want a single control character to conflate these two effects.
>
> Possible reasons why we might not:
>
> a) cases exist in which there is a need to have visual control (e.g. encircling of a sequence) but not determine a unit for functional purposes that affect, for example, collation or "character" count

I am not sure how important character count really is. Does s̏ count as 1 character or 3? Does ŝ̀ with CGJ count as 1, or 2, or 4? It depends how and what you are counting. Counting the absolute number of code points used (for file size) means you can't ignore the zero width or combining characters. But other applications may want to count a° and å both as one character. "How many letters in this word?" is a different question from "How many characters in this document?".

> b) cases exist in which there is a need to have functional control that affect, for example, collation or "character" count, but not visual control
>
> The application that Michael and Ken suggested may relate to (a): if you have

> two combining marks that you want to lay out horizontally over a base rather than stacking them vertically, does insertion of CGJ between them mean that they count as a single unit for purposes of "character" count? One may or may not want that to be the case. And does that mean that the combining marks count as 1 "character" in addition to the base? That question is probably orthogonal to their being layed out horizontally over the base rather than stacking vertically.
>
> Also, if I want a sequence "xy" to be encircled, do I necessarily want them to function as a unit for functional purposes such collation and "character" count? Maybe, but maybe not.

But GJ was not suitable for encirclings, so it's not. Still, is i͡a 1 character or 4? It's 4 in encoding, 3 in semantic units, and 1 in its relation to as a transliteration for Cyrillic я.

> I agree that there is a need for a visual control to deal with the kinds of effects that have been described. I'm also willing to accept the need for a functional control that affects things like collation, though I don't recall for certain that I've seen absolutely convincing arguments. But before we go ahead and conflate these two into a single character, I thought I should at least raise the question of whether we're sure this is what we want to do, and that it won't come back to bite us. It's possible that we'd be better off considering something like a ZERO WIDTH CHARACTER LAYOUT JOINER for the visual control effects, and GRAPHEME JOINER for the orthographic functioning control effects.

I *think* Mark's arguments that ZWJ is suitable for the requirement (because it's already there) outweigh these concerns. But we are still left with Tom's font problem. According to the model discussed, his engine should automatially insert ZWJs to cause the ligatures. Switching fonts to a font without ligatures would then leave these as merely redundant ZWJs with no effect. Users requiring "typewriter" cursive forms while using Tom's font would have to manually delete the ZWJs somehow.

Inputting ص + ح + ح = ; . yields ص + ZWJ + ح + ZWJ + ح , and = صحح . To achieve صحح, the string has to be edited to either ص + ZWJ + ح + ح or ص + ZWJ + ZWNJ + ZWJ + ح + ح.

Decisions with regard to the CGJ still need to be made:

- How does it interact with ZWJ and ZWNJ in terms of font glyph cells?
- Does it impact on Greek or Vietnamese in any way?
- Does it affect the IPA tone contour combinations in any way?