L2/02-363

Response to using ZWJ and ZWNJ to control ligatures

Prepared by: Ayman Aldahleh and Paul Nelson, Microsoft Corporation
Date: 28 October 2002

Overview

In TR #27, section 13.2 Layout Controls was amended to redefine the behavior of the ZERO WIDTH NON-JOINER (ZWNJ – U+200C) and ZERO WIDTH JOINER (ZWJ – U+200D) to control the formation of ligatures. The Indic scripts were exempt from this revision as it significantly contradicted defined usage of these control characters for the Indic scripts.

In TR #28, an additional comment to Ligatures and Latin Typography changed the meaning of the use of ZWJ and ZWNJ to be used only for marking the non-regular cases where ligatures are required or prohibited.

This document identifies manners in which TR #27 breaks existing documents that followed the usage of the ZWJ and ZWNJ as defined prior. It also questions behaviors that are mandated with regards to specific types of font formats to conform to the indicated behavior.

Problems with TR #27

While TR #27 13.2 Layout Controls (revision) seems to be written to solve a problem that exists in a few limited cases, there were shortcomings in the way in which it was written and the information was presented. The specified behavior of TR #27 played down previous assumptions and breaks previously defined behavior for using ZWJ and ZWNJ with the Arabic and Syriac scripts.

Existing definition

TUS 3.0, 13.2 Cursive Connection, The Non-joiner and Joiner (bottom of p. 317), states, “Adding a ZERO WIDTH JOINER between characters that are already cursively connected will have no effect.” TR-27 seems to change this defined behavior. The assumptions based on TUS 3.0 and previous behavior of ZWJ has been useful for many users in resolving Arabic script related issues.

The backing store: ZWJ + ع + ZWJ is demonstrated to be ﻋ in the Unicode book (TUS 3.0, p. 317). Would not ZWJ + ع + ZWJ + ع + ZWJ not be rendered as ﻌﻌ ?

Examples

The first Arabic example in TR-27’s figure to display the new behavior is the Arabic ligature LAM ALEF. The LAM ALEF ligature is a required ligature. It must always form

a ligature. The defined TR-27 behavior breaks default formation of a ligature unless *all* fonts or font engines are modified to the newly defined behavior.

Sample Display Actions

none	ZW NJ	ZW J	ZW NJ	ZW J	ZW J	
لا	ل‌ا	ل‌ا		ل‌ا	لا	TR-27 defined behavior
لا	ل‌ا	ل‌ا		ل‌ا	ل‌ا	Pre TR-27 defined behavior.

When font designers add ligatures to Arabic fonts, they assume that these ligatures will always form without any special markers or higher level protocol involvement. TR-27 comes from the opposite approach saying that the user must specify where ligatures must be placed. Controlling when and where ligatures form is not a widely common scenario, but under the pre-TR-27 behavior users sometimes needed to prevent automatic ligature formation for one of the following reasons:

- Some complex ligatures are hard to read or distinguish at small font sizes, spreading the characters horizontally improves the readability of text.

المحرر المحرر

- Extensive usage of ligatures in Arabic often compresses word's horizontal length considerably, which creates undesirable contrast in length with words that has the same pronunciation weight. Some writers prefer to see these words occupy the same space in the line.

كمجال

لمجال (لمجال)

- When adding diacritics to characters, it makes more sense to have these characters spread horizontally rather than overlapping in one box.

- Finally, for books that teach Arabic, ligatures are not desired. Characters need to occupy their own space.

Today, users of Arabic Windows and Windows applications (which use behavior that is pre-TR-27) have two ways to prevent ligatures from forming. They can add a kashida (tatweel – U+0640) or ZWJ between Arabic characters. ZWJ has the advantage of giving the desired results without adding visible length to words.

TR-27's defined behavior to force ligatures by use of the ZWJ can have serious consequences to user's experience. These behaviors include:

- Existing documents with the hand-controlled scenarios described above will fail to provide the same results. This will also alter the overall look of the documents, including line wrappings and pagination. It is our estimate that the number of documents that are broken is far greater than what TR-27 leads one to believe.
- Today we have many fonts in the Arabic market that makes use of smart font technology to automatically provide creative assortments of Arabic ligatures. The new proposal will force all of them to change their tables to accommodate ZWJ as part of all their ligatures. Even then it will create inconsistency in user experience between existing and new fonts.
- The same point above will apply to applications. Applications might need to change their behavior to allow ZWJ between ligatures, especially with complex cursor movement and selection features. This will cause inconsistency with text appearance in older or simpler applications.

Another item that suggests that using the ZWJ/ZWNJ for ligation control can be seen in that the ZWNJ is supposed to be the manner in which ligatures are prevented. Because the ZWNJ is also used to prevent cursive connection, one is forced to use <ZWJ, ZWNJ, ZWJ> when working with cursive scripts like Arabic and Syriac. This is clearly is a work around that should have indicated a problem with TR-27.

The behavior defined in TR#27 inadvertently adds a canonical decomposition problem. The second Arabic example in the chart shows how MEEM + JEEM (U+0645 + U+062C) can form the ARABIC LIGATURE MEEM WITH JEEM ISOLATED FORM (U+ FC45). The decomposition of this ligature is currently shown as U+0645 U+062C. With the implementation of TR #27, this decomposition must be U+0645 U+200D U+062C to be able to result in the correct representation that these characters should be put into ligature for when they are once again rendered. This has a significant impact on all decompositions in Unicode if changes required by TR #27 are followed.

Non-cursive scripts

Where the ZWJ is useful to force a conjunct or ligature form for linguistic clarity, it should be used…providing it does not change the preexisting definition of the used of the ZWJ/ZWNJ for that language. An interesting question that might be asked is, "If the ZWJ/ZWNJ characters are supposed to be "filtered" out by lexical algorithms, how does one differentiate between "Abcd" and "A<ZWJ>bcd"? If the ZWJ is being used to force

a linguistically required shape, should it also not be required in text handling algorithms to be able to distinguish the difference in plain text?

Putting the burden on font makers

Font vendors are fighting many different issues to make fonts that work. Small software companies like WinSoft (Adobe Middle East versions of PageMaker) and Diwan have significant investments in trying to have their DTP packages comply with Unicode and providing good quality fonts. It would be difficult for small companies like these to go to font vendors and inform them that they need to change their fonts to do something else. I regularly have conversations with Arabic type designers like Mourad Boutros and others who are just trying to stay up with technology and figure out ways to stay in business.

Who are the people paying the font vendors to provide updated fonts to their customers when changes like TR #27 are generated? Does this mean that font vendors like Thomas Milo at DecoType must provide customers with fonts that meet this new standard at no charge? Or, are the small applications developers responsible to bear this burden? In addition, there are many Arabic fonts being used today that have been used for almost 10 years, whose creators are no longer in business. How do those fonts get changed to the new required behavior?

In Egypt and many other Middle East countries, it is not uncommon to find computers that are still running Windows 3.11. TR-27 guarantees that applications generated on these machines cannot be compatible if fonts and applications are updated to some new behavior. TR-27 seemed to give an idea that there were not many documents that used the ZWJ to break ligatures. We suggest just the opposite and hope that the commonly used examples above have sufficiently illustrated this.

Problems with TR #28

TR #28 indicated that it was the task of the rendering system to select ligatures in a manner which was typographically best. It made a statement indicating that the ZWJ and ZWNJ should only be used to force or prohibit ligatures where there linguistically required, “because there are many languages in which this depends not on simple letter pair context but on the meaning of the word in question.”

It appears that TR #28 was attempting to back out where TR #27 had overstep its bounds with regards to specifying how higher level text formatting should work. At the same time TR #28 leaves the possibility of ZWJ and ZWNJ to work for the “many” languages that depend upon the use of ligatures for understandability of the text.

It seems that the assertion of “many” languages requiring ligature forms of characters would provide sufficient examples to draw from and include a real example in the Unicode Book.

Recommendation

In view of the discussions above which illustrates that negative impact of the changes introduced by TR #27, we request that wording in the Unicode 4.0 document be amended to read something like the following:

U+200C ZERO WIDTH NON-JOINER

The intended semantic is to break both cursive connections and ligatures in rendering.

U+200D ZERO WIDTH JOINER

The intended semantic is to enforce a basic connected rendering of adjacent characters that would otherwise be the default displayed by a rendering system. In particular:

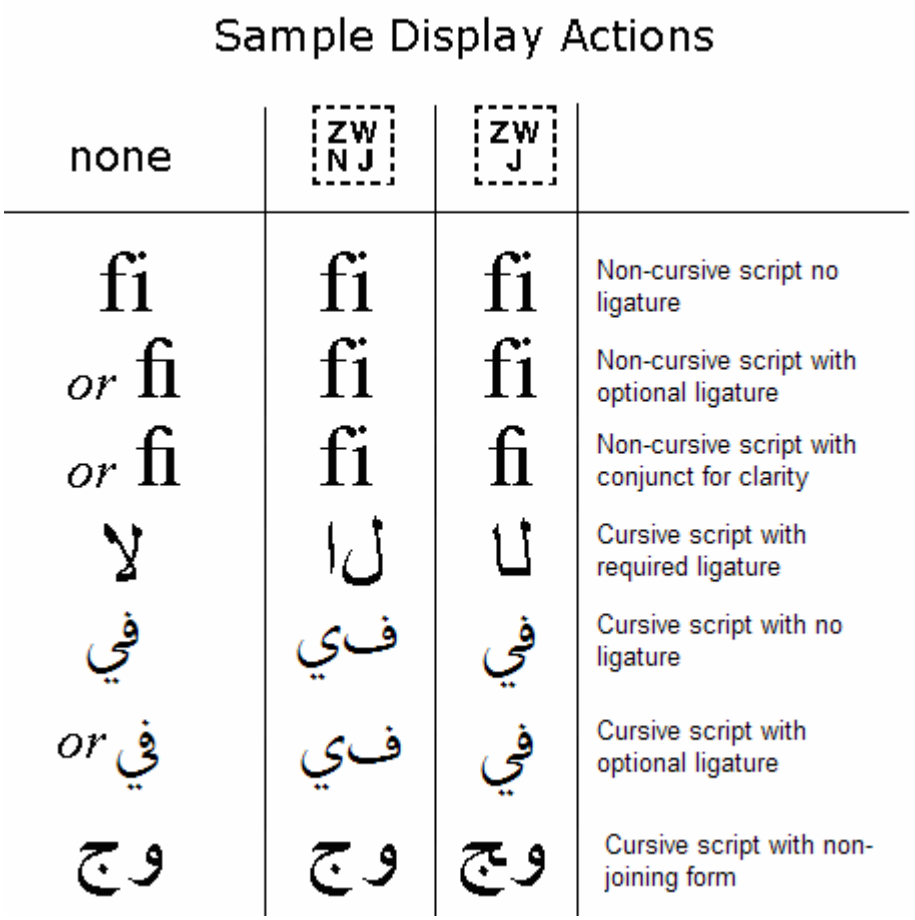
1. In a cursive script, if the two characters could form a ligature by default if the rendering system supports automatic ligatures, ZWJ requests that the ligature not be used and the original joining forms be used instead.
2. In non-cursive script, if the two characters by default do not combine to a conjunct or joined form, and there are possible combinations that should be rendered in plain text, the ZWJ requests that the conjoined form be used. This allows for linguistic distinction where required.
3. In a sequence like <X, ZWJ, Y>, where a cursive form exists for X, but not for Y, the presence of ZWJ requests a cursive form for X.
4. Otherwise, where neither a ligature nor cursive connections are available, the ZWJ has no effect.
5. The ZWJ should not be perceived as a control character that will require the rendering system to select an optional ligature. Typographic forms are controlled by higher level protocols.

The ZWNJ and ZWJ can be used in proximity to each other to override default rendering. For example, if someone wanted to have X take a cursive form but Y to be isolated, then the sequence <X, ZWJ, ZWNJ, Y> could be used (as in previous versions of the Unicode Standard).

Examples of using the ZWJ and ZWNJ are shown in the table below.

Note: Zero width joiner (ZWJ) has a special function when used with Indic scripts. See *Section 9.1, Devanagari*, page 215.

Examples. The following provide samples of desired renderings when the joiner or non-joiner are inserted between two characters. In the Arabic examples, the characters on the left side are in visual order already, but have not yet been shaped. This presumes that all of the glyphs are available in the font. If, for example, the ligatures are not available, the display would fallback to the non-ligature forms.



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.