

**TO: Unicode Technical Committee**

**FROM: Roozbeh Pournader, GNOME Foundation, and Deborah Anderson, SEI, UC Berkeley**

**DATE: 6 August 2009**


**RE: Specifying Joining Behavior for Manichaeian Aleph**

## 1. Background

The Manichaeian script, proposed in L2/09-186 by Michael Everson and Desmond Durkin-Meisterernst, contains an Aleph letter that is not covered by the current Arabic joining model. All other Manichaeian characters – except Aleph – follow the existing Arabic joining model.

Manichaeian Aleph differs from the Syriac Aleph: the Syriac Aleph is a right-joining character that may have minor shape changes at the end of words. In Manichaeian, Aleph has four different contextual shapes, just like dual-joining letters, but its medial and final forms only appear after Beth or Bheth. In other words, it is like a left-joining letter after most letters, but like a dual-joining letter after Beth or Bheth. Compare the following two examples from page 3 of L2/09-186:

 "m'm' =  $\aleph' + \omega m + \aleph' + \omega m + \aleph' + \aleph'$

 "b'b' =  $\aleph' + \beth b + \aleph' + \beth b + \aleph' + \aleph'$

In the top example, the Aleph does not join to preceding Aleph or Mem or the following Aleph, but does join the following Mem (= Aleph Mem\_Aleph Mem\_Aleph Aleph <--- )

In the bottom example, Aleph does not join the preceding or following Aleph but does join the preceding and following Beth. (= Aleph\_Beth\_Aleph\_Beth\_Aleph Aleph <---)

What is the best way to convey this information? Kent Karlsson suggested two options on how to handle Manichaeian Aleph (sent on email over the UnicoRe list, 22 and 23 May 2009), and Roozbeh Pournader offers two additional possibilities. We are requesting UTC to review the options below, and to provide a recommendation.

## 2. Options

**A. New joining types:** Add two new joining types for Manichaeian Aleph, Beth and Bheth, and specify their interaction with existing right-joining, left-joining, and dual-joining types and with themselves.

This appears to be the most straight-forward implication of what Michael Everson and Desmond Durkin-Meisterernst are suggesting in L2/09-186.

Apart from the specified joining behavior, if this option is chosen, Aleph-joining letters will join to their right if they follow either Beth-joining letters (Joining\_Type=B) or join-causing characters (Joining\_Type=C). The purpose of this is to be able to encode stand-alone examples of Aleph in medial and final forms in plain text.

Example data for ArabicJoining.txt:

10AC0; MANICHAEAN ALEPH; **A**; MANICHAEAN ALEPH  
10AC1; MANICHAEAN BETH; **B**; MANICHAEAN BETH  
10AC2; MANICHAEAN BETH WITH 2 DOTS; **B**; MANICHAEAN BETH  
10AC3; MANICHAEAN GIMEL; **D**; MANICHAEAN GIMEL

A variation of this option is to add a new type just for Aleph, but that would require specifying that characters of that joining type would behave differently if they follow characters with specific codepoints (10AC1 and 10AC2) or joining groups (Manichaeon Beth), instead of characters with a specific joining type (B).

**Pros:** No extra burden for text entry. Keeps the existing “one character, one joining type” model.

**Cons:** Would introduce new joining types in the Unicode Standard, which has not happened since the introduction of joining types. Specification of Arabic joining would need to be extended in the text of the standard to allow for the new joining types. All implementations that use a general joining engine for Arabic-like scripts would need to be modified and extended. All parsers of ArabicShaping.txt data file would need to be extended.

**B. Joining types for sequences:** Allow multi-character substrings to have a joining type. Specify that while Manichaeon Aleph, Beth, and Bheth act like normal left-joining and dual-joining letters, the sequences <Beth, Aleph> and <Bheth, Aleph> act like letters themselves with a joining types assigned to the sequence.

Example data for ArabicJoining.txt:

10AC0; MANICHAEAN ALEPH; **L**; MANICHAEAN ALEPH  
10AC1; MANICHAEAN BETH; **D**; MANICHAEAN BETH  
**10AC1 10AC0; MANICHAEAN BETH-ALEPH; D; MANICHAEAN BETH-ALEPH**  
10AC2; MANICHAEAN BETH WITH 2 DOTS; **D**; MANICHAEAN BETH  
**10AC2 10AC0; MANICHAEAN BETH WITH 2 DOTS-ALEPH; D; MANICHAEAN BETH-ALEPH**  
10AC3; MANICHAEAN GIMEL; **D**; MANICHAEAN GIMEL

...

0640; TATWEEL; **C**; No\_Joining\_Group

**0640 10AC0; TATWEEL-MANICHAEAN ALEPH; D; TATWEEL-MANICHAEAN ALEPH**

...

200D; ZERO WIDTH JOINER; **C**; No\_Joining\_Group

**200D 10AC0; ZERO WIDTH JOINER-MANICHAEAN ALEPH; D; ZWJ-MANICHAEAN ALEPH**

(The *tatweel* and ZWJ sequences above are added for this to be functionally equivalent to Option A, in order to be able to encode stand-alone contextual forms of Aleph in plain text.)

**Pros:** No extra burden for text entry. No new joining types.

**Cons:** Lose the existing “one character, one joining type” model. Specification of Arabic joining would need to be extended in the text of the standard to allow for sequences with joining types. All implementations that use a general joining engine for Arabic-like scripts would need to be modified and extended. All parsers of ArabicShaping.txt data file would need to be extended.

**C. Systematic use of ZWNJ:** Identify Aleph, Beth, and Bheth as dual-joining, but advise text encoders to use a ZWNJ character before Aleph whenever it doesn't join a previous letter (i.e., when following any dual-joining or left-joining letter other than Beth and Bheth).

Suggest that input methods insert ZWNJs automatically in such contexts. This would match the existing Arabic model, and the use of ZWNJ would be exactly like many languages written in the Arabic, Syriac, and N'Ko scripts.

Systematic ZWNJ use already exists for some of the orthographies using the Arabic script. This trick is commonly used when the Arabic script is used like an alphabet instead of an abjad, as in Kurdish. In Kurdish, an Arabic Heh (a dual-joining letter) is frequently followed by a ZWNJ to make it act like a right-joining letter.

Example data for ArabicJoining.txt:

```
10AC0; MANICHAEAN ALEPH; D; MANICHAEAN ALEPH
10AC1; MANICHAEAN BETH; D; MANICHAEAN BETH
10AC2; MANICHAEAN BETH WITH 2 DOTS; D; MANICHAEAN BETH
10AC3; MANICHAEAN GIMEL; D; MANICHAEAN GIMEL
```

**Pros:** No new joining types. Keep the existing “one character, one joining type” model. No change to the specification of Arabic joining. No change to implementations that use a general joining engine for Arabic-like scripts. No change to parsers of ArabicShaping.txt data file.

**Cons:** Extra burden for text entry (either typing ZWNJ frequently before Aleph, or using a simple input method).

**D. Obligatory ligatures across non-joining letters.** Identify Aleph as left-joining, but specify that the sequences <Beth, Aleph> and <Bheth, Aleph> would form obligatory ligatures that appear to be joining but really aren't joining. (This is, of course, the most “hack-like” option.)

If this option is chosen, special text should also be added in the standard for <Tatweel, Aleph> (another obligatory ligature), and <ZWJ, Aleph> (not exactly a ligature, since ZWJ does not have a glyph) in order to be able to encode stand-alone contextual forms of Aleph.

This option would be the most troublesome for rendering engines and fonts. For example, rendering engines that justify text by inserting *kashida*'s would need to know that there is a *kashida*-insertion opportunity in the middle of these sequences. Fonts would need to provide extra ligatures, with stop points and *kashida*-insertion information for those, etc.

Example data for ArabicJoining.txt is same as Option C.

**Pros:** No extra burden for text entry. No new joining types. Keep the existing “one character, one joining type” model. No change to parsers of ArabicShaping.txt data file.

**Cons:** Specification of Arabic joining and format control characters would need to be extended to explain the exceptions. Implementations that use a general joining engine for Arabic-like scripts would need to be extended to handle this particular “hack-like” situation properly.

### **3. Recommendation of authors**

The authors consider Option C to be the most conservative solution, requiring minimal changes to the Unicode Standard and its implementations. It's also future-proof: if the existing analysis of Aleph behavior in Manichaean is later found to be inaccurate (e.g., if cases of Aleph joining to a previous letter other than Beth or Bheth are found), Option C provides a way to encode such sequences without changing the standard or its implementations, though some input methods may need to be updated.