L2/16-306

Script property of Arabic Letter Mark and interaction with digit substitution mechanisms

Peter Constable
October 27, 2016

Summary

ARABIC LETTER MARK (U+061C) was added in Unicode 6.3 with a Script property of Arabic, and Script_Extensions of Arabic, Syriac and Thaana. These were changed in Unicode 7.0 to Common. This change is revisited, taking digit substitution implementations into consideration. It is proposed that the original Script and Script_Extensions property values from Unicode 6.3 be restored.

Historical background

ARABIC LETTER MARK ("ALM") was added in Unicode 6.3 with a Script property of Arabic, and a Script_Extensions property of Arab Syrc Thaa. In Unicode 7.0, both property values were changed to Common. That decision was made at UTC #138 (138-C9), based on discussion of L2/13-240. That document proposed a policy regarding relationship between Script and Script_Extensions property values. It also discussed ALM specifically, suggesting that there was no for an explicit script property:

> "ALM should become sc=Common, scx={Common}. There's no need for it to specify script(s). It was only encoded in the Arabic block to get a default bidi class of AL. A gratuitous differentiation from other bidirectional controls, which are all sc=Common, scx={Common}, adds to the confusion partially created by its name and block. The character is not at all restricted to those scripts in usage, and if used with other scripts, should not trigger shaping engines to switch to a different layout engine which could be very disruptive."

This rationale appears to have been the driving factor that led to the property changes for ALM in Unicode 7.0.

A separate, related outcome from L2/13-240 was 141-C10, adoption of the following policy:

> Where a character's Script_Extensions value set has more than one element, the character's Script value must not be explicit, *except* where that script is a reasonable default value.

Interaction with digit substitution

ALM was first proposed in L2/11-005 as work was happening on bidi isolates, and was initially motivated by limitations of RLM (U+200F) in regard to bidi-ordering effects on digits in Arabic contexts:

> "The UBA specifies that Arabic letters form an Arabic context wherein following Arabic-European digits must be handled as Arabic-Indic digits, but the presence of an RLM, which may be needed for ordering reasons, destroys this context."

But L2/11-005 also goes on to mention an interaction with digit-substitution behaviours:

> "In addition, there is a need to transform Arabic-European digits into Arabic-Indic digits when these digits are positioned at the start of the text like in formulas and numbered lists."

Digit-substitution is a display-time process that causes characters in text that are digits in the Basic Latin block to be displayed as digits from other scripts to match user cultural preferences. Digit substitution mechanisms have a long history, as suggested by the inclusion in Unicode since version 1.1 of legacy control characters U+206E NATIONAL DIGIT SHAPES and U+206F NOMINAL DIGIT SHAPES.

In systems that use digit-substitution, it typically is one of the various settings associated with a "locale": some locales specify no substitution, while other locales specify substitution with "native" / "traditional" digits — i.e., digits associated with the script used in that locale. In some systems, certain locales may have a third setting: "contextual". With contextual digit substitution, digits are or are not substituted based on context in which they occur. Specifically, the way that digits are display is based on the preceding text in the string: if preceded by Latin text, they are not substituted, but if preceded by text in the script used in that locale, then they are substituted to the native digits.

It is important to note that contextual substitution is only ever used in locales that use Arabic script.

While ALM was encoded in Unicode 6.3, it does not appear to have been used much. Recently, however, it has been incorporated into number formatting patterns in CLDR 30.

> “Number symbols (plus, minus, percent) and formats (currency, percent) were updated for ar, fa, he in an effort to produce better results. Note that depending on locale and numbering system, those number symbols and currency patterns may contain bidi controls LRM, RLM, and ALM. Use of the ALM is new in CLDR 30…”

For instance, the following was introduced in ar.xml:

```
<percentSign>٪؜</percentSign> <!-- includes ALM after sign -->
<plusSign>؜+</plusSign> <!-- includes ALM before sign -->
<minusSign>؜-</minusSign> <!-- includes ALM before sign (002D) -->
```

Thus, ALM can occur as the first preceding strongly-directional character — and, in some contexts, the only strongly-directional character — before a formatted number.

With this in mind, let us return to this statement in L2/11-005 regarding requirements for encoding ALM:

> "In addition, there is a need to transform Arabic-European digits into Arabic-Indic digits when these digits are positioned at the start of the text like in formulas and numbered lists."

This indicates that the requirement was to provide not only a strongly-directional control character with bidi class AL, but also an explicit-script character to induce digit substitution for Arabic-script locales.

Contextual digit substitution implementations will look for the first preceding explicit-script characters and base the condition for substitution on the script of that character. With ALM having a script property value of Arabic, as when first encoded in Unicode 6.3, the requirement stated in L2/11-005 is met. With the script property changed to Common in Unicode 7.0, however, no substitution will occur.

Script extensions for ALM

As noted above, ALM was initially assigned a Script property of Arabic, and a Script_Extensions property that included Arabic, Syriac and Thaana. The policy adopted in 141-C10 has a bearing on such property assignments in that they comprise an exception to the policy: given the Script_Extensions value, the explicit Script property value is permitted under that policy only if that script is a reasonable default.

ALM was created for use in the context of right-to-left scripts that have characters with a bidi class of AL. In Unicode 9.0, the set of such scripts is limited to Arabic, Syriac and Thaana. Use of AL in those scripts is the reason for the originally assigning those scripts as script extensions for ALM.

It is also important to note that these scripts do not have their own native digits, and the cultures that use these scripts use Arabic-Indic digits. For purposes of digit substitution, then, it would be appropriate to treat this character as Arabic. And since it is a default-ignorable control character, the only common text process in which the script property has particular significance is digit substitution.

On that basis, for ALM to have an explicit script property of Arabic while having multiple script extensions seems to be reasonable since, in all contexts in which it is likely used, Arabic script is a reasonable default.

Proposal

This motivation given in in L2/13-240 for changing the Script and Script_Extensions properties of ALM was:

> “There's no need for it to specify script(s). It was only encoded in the Arabic block to get a default bidi class of AL. A gratuitous differentiation from other bidirectional controls, which are all sc=Common, scx={Common}, adds to the confusion partially created by its name and block. The character is not at all restricted to those scripts in usage, and if used with other scripts, should not trigger shaping engines to switch to a different layout engine which could be very disruptive.”

While the Arabic block might have been chosen simply as an easy way to inherit the desired bidi class of AL, the points raised above call into question the assertion that “There’s no need for it to specify script(s)”: As has been shown, an explicit script assignment facilitates desired results for contextual digit substitution, which was one of the originally-stated requirements for encoding ALM in L2/11-005. And, contrary to the assertion that the explicit script assignment could be disruptive, it is the very lack of an explicit script assignment after the changes in Unicode 7.0 that disrupts desired behavior in digit substitution.

Based on the above considerations, I propose that the Script and Script_Extensions property values for ALM be restored to the original values assigned in Unicode 6.3.