Response of IP to Malayalam Draft LGR of 2018-08-29

DATE: 2018-09-12

# Overview

This document provides IP response to the Malayalam LGR proposal dated 2018-08-29.

All of the previously raised issues appear to have been accommodated. The LGR for Malayalam therefore appears ready for public comment after fixing some minor issues in DOCx file and correcting the use of context rules for one of the sequences in the XML file (see below for discussion).

# General Comments

|  |  |  |
| --- | --- | --- |
| **Item** | **Issue** | **IP Comment** |
|  | The IP made a number of editorial/formatting suggestions for the Malayalam LGR document. They are provided in a separate marked-up document. (Note that formatting changes are too numerous and have not been change tracked individually: the generally aim at consistent and uniform use of font family, font size and line spacing throughout) | Please review and make any other changes as needed |
| 0D7F | This chillu appears to be rather rare; this is mentioned in the Unicode Standard, and also shows by being absent from some of the text corpora consulted by the IP.  The IP has no objection to including this code point – it would seem that treating the chillus consistently is a good policy, and presumably, the code point is in modern use, except that it occurs more rarely.  However, the GP may want to consider adding a sentence pointing out that the code point was deliberately added (e.g. because of modern use and/or consistency for treating chillu), even though it happens to be rare. | Please consider adding such an explanation. However, as this suggestion is optional, the IP has not provided suggested language in the marked-up DOCx file. |

# XML

|  |  |  |
| --- | --- | --- |
| **Item** | **Issue** | **IP Comment** |
|  | The IP reviewed the use of context rules for the sequence U+0D28 U+0D4D U+0D31 suggested by previous IP feedback.  After more thorough review, the IP comes to the conclusion that instead of adding a context rule on the sequence U+0D28 U+0D4D U+0D31, the context rule needs to be added to the variant mapping to/from U+0D7B U+0D31.  The existing context rule for U+0D7B U+0D31 is correct.  These changes affect only the XML file.  The changes are: | **These fixes have been applied to the attached XML file** |
| **line 206** | Remove not-when=”follows-B-X-or-H” from <**char** cp=”0D28 0D4D 0D321” .. > |
| **line 207** | Add not-when=”follows-B-X-or-H” to <**var** cp=”0D7B 0D31” … > |
| **line 248** | Add not-when=”follows-B-X-or-H” to <**var** cp=”0D28 0D4D OD31” …. > |
| **<description>** | In the description there's one sentence that the GP marked as "unchanged" in their response to the latest IP feedback. The sentence is  "Each code-point has associated Glyph, Character Name, ~~Unicode General Category (gc),~~ Indic Syllabic Category and References."  where the words "Unicode General Category (gc)" should be deleted as shown above by strikeout. (General category information is not actually listed in the XML - and shouldn't be -- therefore the statement needs to be corrected as indicated). | **This fix has been made in the attached XML** |

# Test files

|  |  |  |
| --- | --- | --- |
| **Item** | **Issue** | **IP Comment** |
|  | no issues |  |

# Appendix on Further Review of Variant Issues and Context Rules

The following analysis shows the effect of the proposed changes in lines 206, 207 and 248 of the XML.

## (1) LGR as Proposed 2018-08-29

In the following analysis, we assume that these mappings have been defined for blocked variants:

|  |  |  |
| --- | --- | --- |
| Variant Set 1 | 0D7B 0D31 🡨🡪 0D28 0D4D 0D31 | no context |

Also assumed that a context rule is defined for both of the code point sequences:

0D28 0D4D 0D31: not-when(follows-B-X-or-H).  
07DB 0D31: not-when(follows-B-X-or-H).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Label | Partition | Disp | Variant | Disposition |
| 0D7B 0D31 | {0D7B}{0D31} | valid | 0D7B 0D31 | valid |
| {0D7B 0D31} | valid | 0D7B 0D31 | valid |
| 0D28 0D4D 0D31 | valid |
| 0D28 0D4D 0D31 | {0D28} {0D4D} {0D31} | valid | 0D28 0D4D 0D31 | valid |
| {0D28 0D4D 0D31} | valid | 0D28 0D4D 0D31 | valid |
| 0D7B 0D31 | valid |
| .. X 0D7B 0D31 | {0D7B}{0D31} | *invalid* | ..X 0D7B 0D31 | *invalid* |
| {0D7B 0D31} | *invalid* | ..X 0D7B 0D31 | *invalid* |
| ..X 0D28 0D4D 0D31 | **valid** |
| ..X 0D28 0D4D 0D31 | {0D28} {0D4D} {0D31} | valid | ..X 0D28 0D4D 0D31 | valid |
| {0D28 0D4D 0D31} | *invalid* | ..X 0D28 0D4D 0D31 | valid |
| ..X 0D7B 0D31 | ***invalid*** |

where …X is a placeholder for some context ending in X (and thus violating the context rule).

Note that there is no context constraint on 0D28 or 0D31 and the context constraints on 0D4D is compatible with it occurring following 0D28. For that reason, the *partition*  {0D28} {0D4D} {0D31} — where each of the code points are treated as singletons — is not restricted and fully valid, even if preceded by a sample context “..X”. The rule here is that a label is invalid if ALL of its partitions are invalid; conversely, a label is valid if ANY of its partitions are valid.

Validity testing of a *variant label* is performed separately following the permutation that substitutes each code point variant (or variant sequence). Therefore, the specific partitioning that lead to the variant label is not retained and the disposition is determined as if it were an original label. As a result, several of the rows in the Variant column have multiple partitions and therefore do not result in invalid variant labels. In one case, an invalid label could formally have a valid variant.

Despite the context rule on the sequence <0D28 0D4D 0D31> the LGR generates invalid variants that are not simply the unpermuted originals for an invalid label (last line).

We also conclude (see “Note” above) that the context rule does not constrain the validity of original labels, leaving it out, would therefore not change the LGR.

## (2) LGR as modified according to this feedback:

In the following revised analysis, we assume that these mappings have been defined for blocked variants and that they are subject (on the variant level) to a context rule:

|  |  |  |
| --- | --- | --- |
| Variant Set 1 | 0D7B 0D31 🡨🡪 0D28 0D4D 0D31 | not-when(follows-B-X-or-H). |

Also assumed that a context rule is defined for one of the code point sequences, but no both:

0D7B 0D31 : not-when(follows-B-X-or-H).

The latter context rule is required so that sequence <0D7B 0D31> does not override the context constraint defined for 0D7B. Unlike the case in (1), the omission of this context rule would change the LGR.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Label | Partition | Disp | Variant | Disposition |
| 0D7B 0D31 | {0D7B}{0D31} | valid | 0D7B 0D31 | valid |
| {0D7B 0D31} | valid | 0D7B 0D31 | valid |
| 0D28 0D4D 0D31 | valid |
| 0D28 0D4D 0D31 | {0D28} {0D4D} {0D31} | valid | 0D28 0D4D 0D31 | valid |
| {0D28 0D4D 0D31} | valid | 0D28 0D4D 0D31 | valid |
| 0D7B 0D31 | valid |
| .. X 0D7B 0D31 | {0D7B}{0D31} | *invalid* | ..X 0D7B 0D31 | *invalid* |
| {0D7B 0D31} | *invalid* | ..X 0D7B 0D31 | *invalid* |
| ..X ~~0D28 0D4D 0D31~~ | DOES NOT EXIST |
| ..X 0D28 0D4D 0D31 | {0D28} {0D4D} {0D31} | valid | ..X 0D28 0D4D 0D31 | valid |
| {0D28 0D4D 0D31} | VALID | ..X 0D28 0D4D 0D31 | valid |
| ..X ~~0D7B 0D31~~ | *DOES NOT EXIST* |

where items in UPPERCASE denote results different from (1) above.

Note that per RFC 7940 a context on a variant mapping treats the variant as “not defined” when the context requirement is not met. Therefore, adding the context rule to the variant removes the case where a “valid” original label can create an “invalid” variant.

Any “invalid” items in the variant column are original (unmodified) versions of an invalid label and that would be as expected.

## Conclusion

As can be seen from this analysis, removing the context rule on <0D28 0D4D 0D31> and adding the same context rule on the variant definitions results in the expected behavior that valid labels create only valid variants and invalid labels do not create valid variants.