Universal Multiple-Octet Coded Character Set
International Organization for Standardization

Doc Type: ISO/IEC JTC1/SC2/WG2/IRG
Title: Proposal to correct inconsistent total strokes data in the Unihan database
Authors: Ken Lunde
Status: Member Body Contribution
Action: For consideration by the IRG
Date: 2022-09-06

The background of this proposal is that the Unicode Consortium received the following public feedback on 2022-03-12:

> I found out an issue in Unihan Database. Some kTotalStrokes values of the characters with the radical 邑 or 阜 may be wrong. For example, kTotalStrokes value of U+2B545 𫩅 is 10, while U+2CBC0 𬯀 is 9. The radical 阝 has 2 strokes in the blocks from CJKUI to CJK-ExtD, while it has 3 strokes in the blocks from CJK-ExtE to CJK-ExtG. I wonder whether this is wrong. In the other words, the stroke of 阝 is 3 since Unicode® 8.0.0 was published.

I was subsequently tasked by the UTC (*Unicode Technical Committee*) per UTC #171 Action Item 171-A36 to prepare a proposal for the IRG #59 meeting, which is the sole purpose of this document.

The problem

As the public feedback dated 2022-03-12 clearly indicates, there is an inconsistency between the *kTotalStrokes* property values of earlier CJK Unified Ideographs blocks, specifically the URO and Extensions A through D, and those of later blocks, specifically Extensions E through H. This inconsistency is manifested in particular radicals, such as the 阝 form of Radicals #163 (邑) and #170 (阜), both of which should be counted as three strokes, but are often counted as two strokes. My own cursory checking revealed that this issue is also manifested in other radicals, such as Radicals #140 (艹/艹/⺾) and #162 (辶/辶/辶), both of which should be counted as four strokes, but are often counted as three strokes. I additionally observed that these inconsistencies tend to vary by block, and is therefore not actually manifested in earlier versus later blocks. In other words, there is simply an inconsistency problem.

Radical #188 (骨/骨) is a special case whereby most ideographs that include this radical should ideally specify two *kTotalStrokes* property values that reflect 9 and 10 as number of strokes when used either as the radical or as a component, particularly when an ideograph includes both a G- and T-Source source reference.

As of Unicode Version 15.0 (2022), there are now 97,058 CJK Unified Ideographs among the nine blocks: the URO and Extensions A through H. The UTC has been adjusting *kTotalStrokes*—and to a similar extent, *kRSUnicode*—property values largely on a one-off basis, based on er-

ror reports received as public feedback. In order to find a solution to this problem, which is systemic in nature, and when considering the sheer number of potentially affected property values, a programmatic solution is therefore necessary.

The *kTotalStrokes* property values are used for collation purposes, so it is important to have the most accurate property values as possible. A similar statement can be made for *kRSUnicode* property values. Both properties have 100% coverage in the Unihan database.

A proposed solution

I believe that a viable solution to this problem can be found in one of the features of the ORT (*Online Review Tool*) that the IRG uses to review IRG working sets. The excerpt below is from IRG Working Set 2021 #04254, and demonstrates that IDSes can be used to programmatically calculate the number of total strokes of an ideograph:

Attributes:

Char	SC	FS	TC
𠚪 (U+961D)	N/A	N/A	3
于 (U+4E8E)	3	1	3

	Expected	Recorded	
SC	3	3	OK
FS	1	1	OK
TC	6	6	OK

Of course, there are edge cases that may require manual intervention, but at least this solution should cover the vast majority of cases.

In terms of executing the solution, I propose that the ORT Manager—or someone who is familiar with the inner workings of the ORT—be tasked to apply the same IDS-based total strokes calculations against all 97,058 ideographs in the nine CJK Unified Ideographs blocks, and the results could then be compared with the current *kTotalStrokes* property values. Multiple IDS databases could be leveraged, which, when their results are combined, may reveal additional inconsistencies or handle particular edge cases. This exercise is also likely to reveal a non-zero number of issues in current *kRSUnicode* property values.

Of course, we need to be clear that the end result of this exercise should not be expected to result in perfect *kTotalStrokes* property values, but it should at least resolve the vast majority of the inconsistencies that are present in the current property values. The UTC can then continue to handle additional property value corrections on a one-off basis.

That is all.