L2/08-373

ISO/IEC JTC1/SC2 WG2 *N3525*

TITLE: Handling CJK compatibility characters with variation sequences
SOURCE: Ken Lunde and Eric Muller, Adobe Systems
STATUS: Expert contribution
ACTION: For consideration by WG2
DISTRIBUTION: ISO/IEC JTC1/SC2/WG2
DATE: 6 October 2008

Handling CJK compatibility characters with variation sequences

Ken Lunde, Adobe Systems Inc.
Eric Muller, Adobe Systems Inc.

October 6, 2008

1. Introduction

The canonical decomposition of CJK compatibility ideographs is problematic, as it "erases" distinctions in source standards which need to be preserved for proper round-tripping and rendering. This situation goes beyond the need for locale specific forms, as those distinctions are needed within a single locale.

The purpose of this contribution is to show how variation sequences can be used to retain the distinctions, even if the data is normalized.

Throughout this paper, the term "compatibility ideograph" refers to the canonical decomposable characters encoded in the CJK Compatibility Ideographs block and the CJK Compatibility Ideographs Supplement block. The twelve characters in those blocks which are not decomposable are covered by the term "unified ideographs", along with the characters in the CJK Unified Ideographs block, the CJK Unified Ideographs Extension A block and the CJK Unified Ideographs Extension B block.

2. Some examples

Here are a couple of examples that will be used to illustrate the various situations.

Example 1: U+65E2, and the compatibility ideograph that normalizes to it, U+FA42.

Example 1

code point	Unicode 4.0 glyph	Source	ISO 10646-1:2000 or -2:2001 glyph
U+65E2		G0-3C48	
		J0-347B	

code point	Unicode 4.0 glyph	Source	ISO 10646-1:2000 or -2:2001 glyph
		T1-514D	
U+FA42		J3-752B	

This is the simplest case: the same glyph shape is present in three of the source standards that contribute to the URO, they are unified as U+65E2. Later, JIS 0213 is added; the glyph shape for J3-752B is unified with the original one, but to provide round-tripping to JIS 0213, U+FA42 is encoded to maintain the distinction between J3-752B and J0-347B.

Note that the fact that the two shapes are unified mean that in the absence of other information. either shape is an appropriate rendering for either character,

Example 2: U+4FAE, and the two compatibility ideographs that normalize to it, U+FA30 and U+2F805:

Example 2

code point	Unicode 4.0 glyph	Source	ISO 10646-1:2000 or -2:2001 glyph
U+4FAE		G0-4E6A	
		J0-496E	
		KP0-5932	
		K0-5932	
		T1-4F78	
U+FA30		J3-2E38	
U+2F805		KP1-3534	
		T4-253D	

What we have here are two glyph shapes: let's call them the G shape (oblique strokes) and the Jo shape (vertical

stroke). According to the unification rules, those two shapes represent the same abstract character.

Thus, when the URO was built, the source characters G0-4E6A, J0-496E, K0-5932 and T1-4F78 were encoded as a single coded character, U+4FAE.

When Extension B was built, the T4 source was added. Because (T1, T4) is not subject to the source separation rule, T4-253D did not result in a separately encoded character. But to maintain round-tripping with the TCA-CNS 11643-1992 standard (the various planes of it are the T sources), the compatibility ideograph U+2F805 was encoded. (Note that round-tripping with CNS 11643-1992 is actually not provided in general; presumably, this is done one a character by character basis, and this case was deemed worthy of a compatibility ideograph.)

Finally, the J3 and J4 sources were added. Again (J0, J3) is not subject to the source separation rule, but round-tripping with the combined JIS 0208/0213 standard (the J0, J3 and J4 sources), the compatibility ideograph U+FA30 was encoded.

It is not known to the authors when the KP sources were added.

Note that it may have been nice to invert the J sources, that is to give J3-2E38 as the source for U+4FAE, and J0-496E as the source for U+FA30. This would have made the preferred glyph shape for U+4FAE the same in all locales. However, this was probably prevented by stability considerations.

Also, the encoding of U+2F805 would have been superflous, had the compatibility ideographs been approached in a multi-column fashion: U+FA30 can perfectly be used contrastively with U+4FAE to maintain the distinction in the KP and T sources, just like this pair is used to maintain the distinction in the J sources.

3. The problem

Consider what happens for an implementation that consumes and produces JIS 0208/0213 characters, and uses the UCS to represent characters internally. The input J0-347B is represented internally by U+65E2; the input J3-752B is represented internally by U+FA42. Using a font appropriate for Japanese, the correct glyphs can be displayed. On output, the appropriate JIS characters can be generated. So far so good.

If normalization occurs at some point during processing, U+FA42 is normalized to U+65E2. From then on, it is no longer possible to properly render or output in JIS. All the efforts by 10646 and by the application to ensure round-tripping with JIS 0208/0213 are simply negated.

The normalization may be performed by a component of the application over which the author of the application has little or no control. In other words, “do not normalize” may not be a viable option. Excluding the CJK ideographs from normalization is not a viable option for the same reason.

4. Proposed solution

To solve the problems above, the idea is to transform each compatibility ideograph into something that is still distinct from the corresponding unified ideograph, yet is not reduced to it by normalization. A possible choice would be a PUA code point. A better choice is a variation sequence involving the unified ideograph, as the (10646) identity of the character is retained and is accessible to any component or subprocess of the application, without the need for a private agreement.

Assuming that the transformation can be performed before any normalization occurs, may be as soon as inputs are converted from JIS to the UCS, then the distinction is preserved. The inverse transformation can be applied after any normalization may occur, may be as late as when the output is converted from the UCS to JIS. In fact, the transformation and its inverse can be integrated in the JIS / UCS conversions.

The proposal is actually to define one variation sequence for each compatibility ideograph, as well as one for the corresponding unified ideographs, and to define the variation sequences by the source characters. Using our first example:

Table 1

source standard	today's conversion	“normalization-safe” conversion	glyphs
G0-3C48, J0-347B, T1-514D	<65E2>	<65E2c VS1>	
J3-752B	<FA42>	<65E2, VS2>	

Note that we do define a variation sequence for the unified ideograph. The motivation is this: it is perfectly acceptable for a font to use the glyph shape of J3-752B to render U+65E2 alone; so when we want the glyph shape of J0-347B, we cannot use U+65E2 alone.

Also, in the examples above, VS1 and VS2 can be any two variation selectors (as long as there is no conflict with the variation sequences already present in the IVD).

For our second example, the similar creation of one variation sequence for each code point leads to:

Table 2

source standard	today's conversion	“normalization-safe” conversion	glyphs
G0-4E6A, J0-496E, KP0-5932, K0-5932, J0-496E, T1-4F78	<4FAE>	<4FAE, VS1>	 or
J3-2E38	<FA30>	<4FAE, VS2>	
KP1-3534, T4-253D	<2F805>	<4FAE, VS3>	

However, such of use of variation selectors does not fit very nicely with the notion that a variation selectors restricts the range of glyphs which can be use to render a character, as can be seen with the first sequence. This is mostly because of the historical order in which the sources have accomodated in the UCS, as we detailed earlier. We can rectify this by using this mapping instead:

Table 3

source standard	today's conversion	“normalization-safe” conversion	glyphs

source standard	today's conversion	“normalization-safe” conversion	glyphs
G0-4E6A, KP0-5932, K0-5932, T1-4F78, **J3-2E38**	<4FAE> or <FA30>	<4FAE, VS1>	
J0-496E, KP1-3534, T4-253D	<4FAE> or <2F805>	<4FAE, VS2>	

Both approaches are effective to solve the normalization problem. The second approach (table 3) has the advantage that it uses variation selectors exactly for their intented purpose, and it is probably what would be done should the UCS be created today, knowing all the sources that have to be accomodated. It has the disadvantage that the conversion from today's UCS representation to the normalization-safe representation depends on the source of the material to be converted: <4FAE> has to be transformed to either <4FAE, VS1> or <4FAE, VS2> depending on whether this <4FAE> is the result of a conversion of a G, KP, K, or T source or is the result of a conversion of a J source. If the conversion of non-UCS data to UCS is done directly to the normalization-safe representation (i.e. from the first column directly to the third column), this slight complication disappears entirely.

Before recommending one approach, let's look at another situation.

5. Handling source-based glyph variations

If we look at the solution proposed in table 3, it can be interpreted at taking all the glyphs shapes for U+4FAE (including those for its compatibility ideographs U+FA30 and U+2F805) in the source standards, and creating variation sequences as necessary to distinguish those shapes. This approach is applicable to situations that do not involve compatibility ideographs, as for example U+50A6:

Example 3

code point	Unicode 4.0 glyph	Source	ISO 10646-1:2000 or -2:2001 glyph
U+50A6		GE-2233	
		T3-644D	
		V0-3037	

It is natural to define the following variation sequences:

Table 4

source standard	today's conversion	“normalization-safe” conversion	glyphs
GE-2233	<50A6>	<50A6, VS1>	

source standard	today's conversion	“normalization-safe” conversion	glyphs
T3-644D, V0-3037	<50A6>	<50A6, VS2>	

Having such variation sequences for all unified ideographs, not just those which are the decomposition of compatibility ideographs, would solve a long standing problem in the use of the UCS: today, a plain text representation, made solely of <50A6>, is not enough to ensure that the display of that plain text will be acceptable to all readers: it is necessary to use “out-of-band” (i.e. non plain text) mechanism to ensure that the first or the second glyph shape will be used.

6. Implementing the solution

The implementaion of the solution proposed above involves a number of separate decisions.

Where?

> The first decision is how to standardize the variation sequences. One possibility is to add new sequences to the 10646 clause that lists all the acceptable variation sequences. The other possibility is to register the sequences in the Ideographic Variation Database. We feel that this approach would be much more appropriate, as it would put all the variation sequences related to CJK ideographs in a single place.

Which sequences?

> In section 4, we presented two approaches for U+4FAE. We believe that the second approach (Table 3) is vastly preferable to the first approach (Table 2) because it is better aligned with the notion of variation selector, and because it can be extended to cases like U+50A6. It involves a bit more work, as the sequences cannot be determined mechanically from the source data (one has to inspect the glyph shapes in the charts), but it is well worth the extra effort.

Stages?

> While we advocate the second approach should be followed, and that sequences should eventually be defined for the full repertoire, it would be perfectly appropriate to conduct this work in stages. Those unified ideographs which have compatibility equivalent are the obvious candidates for a first stage, as this would take care of a situation that is fundamentally broken today. The amount of work for this stage would be moderate, as it concerns 898 ideographs. The extension to the other unified ideographs could be done later, may be in multiple stages.

Assuming our recommendations, this would lead to a registration of a new collection. The name of the collection could “Sources” or “IRG”. There is no completely obvious choice for the identifiers in that collection, so as a strawman proposal, we recommend something based on the sources.

For the two examples we have used above, this would lead to the following sequences being registered in a first stage:

```
4FAE E0102; Sources; G0-4E6A_KP0-5932_K0-5932_T1-4F78_J3-2E38
4FAE E0103; Sources; J0-496E_KP1-3534_T4-253D

65E2 E0102; Sources; G0-3C48_J0-347B_T1-514D
65E2 E0103; Sources; J3-752B
```

(The particular sequences listed above are not yet used in the IVD as of the writing of this document, and would be the next ones to be assigned; should another registration be made before the one proposed here, it could be that other variation sequences are selected.)

In subsequent stages, we would have sequences such as:

```
50A6 E0100; Sources; GE-2233
50A6 E0101; Sources; T3-644D_V0-3037
```

7. Interaction with existing IVD sequences

In the case of U+4FAE, we already have two sequences registered in the IVD, as part of the Adobe-Japan1 collection:

sequence	Adobe-Japan1 identifier	glyph
<4FAE, E0100>	CID+3552	
<4FAE, E0101>	CID+13382	

So there are two possibilities for this case:

- register variation sequences as outlined above, independent of the two Adobe-Japan1 sequences.
- for U+4FAE, just reuse the two variation sequences of Adobe-Japan1.

Either approaches would serve the primary purpose. There will probably be some cases where Adobe-Japan1 makes more distinction than are necessary for the problem at hand.