Accredited Standards Committee* **NCITS, Information Processing Systems**	Doc:	L2/98-057
	Date:	Date:
	Project:	Project:
	Reply To:	Peter Edberg Apple Computer, Inc. 2 Infinite Loop, MS: 302-2IS Cupertino, CA 95014 Tel: 408 974-4275 Fax: 408 862-4566 E-mail: pedberg@apple.com

*Operating under the procedures of the American National Standards Institute
NCITS Secretariat, Information Technology Industry Council
1250 Eye Street NW, Suite 200, Washington, DC 20005-3922, Tel: 202 737-8888, Fax: 202 638-4922

Subject: Proposal for transcoding hint characters (version 2)

A. Introduction & Problem Description

Often there is an engineering requirement for perfect roundtrip conversion from legacy text data in various character encodings to Unicode text and back. This can occur when converting file system catalogs, for example.

The legacy text data may be in encodings which are not among the set for which the Unicode Standard currently provides guaranteed roundtrip fidelity. Some characters in these encodings may be subject to one of the following problems:

1. No corresponding Unicode character (or sequence of Unicode characters)

In this case, the legacy character is usually be mapped to a Unicode private use character. Text content will be lost when the resulting Unicode is interchanged, but this is unavoidable; there is no way to represent the legacy character in Unicode. Fortunately, this problem is rather rare.

Example 1: fleur-de-lis, Mac OS Korean (EUC-KR ++) and others

Mac OS Korean
0xA642 ——— Unicode
(no standard character,
use private-use character)

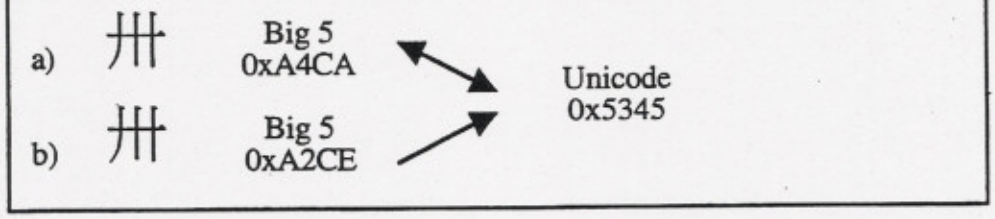
2. The corresponding Unicode character (or sequence) conflicts with the mapping for another character or sequence of characters in the legacy encoding

This is much more common than problem 1 above, and can be addressed using the techniques described in this proposal.

A legacy encoding may have duplicate encoding of a character that is only encoded once in

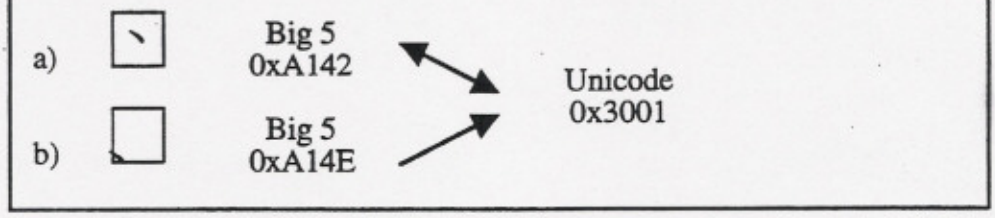
Unicode. In the following Big 5 example, part b shows a character in the Hangzhou numeral block that duplicates a character in the main CJK ideograph block (shown in part a). The corresponding Unicode character is mapped back to the character in part a, so the character in part b has no roundtrip mapping.

Example 2: duplicate CJK ideograph thirty (Hangzhou numeral), Big 5



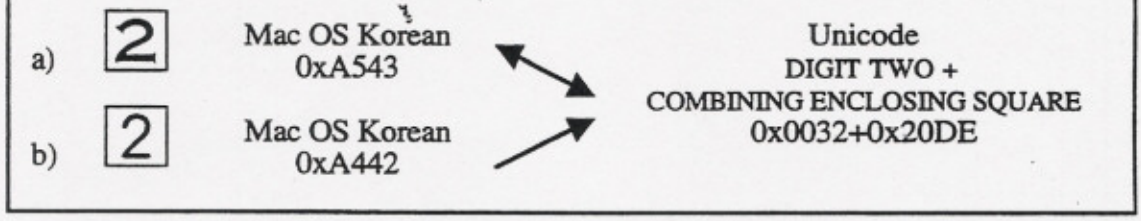
A legacy encoding may have multiple presentation forms that all correspond to a single Unicode character. In the following Big 5 example, the character in part a is the form of IDEOGRAPHIC COMMA used for vertical text; this has a roundtrip mapping to Unicode. The character in part b is the form used for horizontal text; it maps to the same Unicode character and thus has no roundtrip mapping.

Example 3: presentation form for IDEOGRAPHIC COMMA, Big 5
(forms for vertical and horizontal text, shown in CJK "character cell")



A legacy encoding may have multiple presentation forms that all correspond to a single Unicode sequence. This is similar to example 3, but the Unicode mapping is a sequence.

Example 4: presentation form for digit two in enclosing square, Mac OS Korean



A legacy encoding may include multiple spellings of a given text element, but these might all correspond to a single Unicode sequence. Mac OS Japanese includes a single character for Roman numeral fourteen, which it can also represent using a character sequence. Unicode has single characters for Roman numerals one through twelve (direction neutral), but Roman numeral fourteen can only be represented in Unicode as a character sequence (direction LR).

Example 5: Roman numeral fourteen, Mac OS Japanese (Shift-JIS ++) and others

a) X+I+V — Mac OS Japanese 0x58+0x49+0x56

b) ⅩⅣ — Mac OS Japanese 0x85AC

Unicode 0x0058+0x0049+0x0056

In examples 2-5, the legacy characters in part b could also be mapped to private use characters for roundtrip fidelity, with the corresponding loss of interchangeability for the resulting Unicode text. This would be unfortunate, because the content of the legacy characters can be represented in Unicode. A better solution is to map the legacy characters in part b of these examples to the standard Unicode characters as shown, plus a transcoding hint character that allows them to be distinguished for roundtrip fidelity. This document proposes a set of such hint characters.

B. Proposal

The proposal is to add two small sets of transcoding hint characters to the BMP:

1. Add a set of 16 "transcoding variant hint" characters (0xNN00-0xNN0F)

A "transcoding variant hint" is like a combining character. It can follow a base character, or it can follow a sequence of characters consisting of a base character followed by standard combining characters. The entire sequence consisting of Unicode base character plus standard combining characters (if present) plus variant hint is considered to be a single element for transcoding. A variant hint always terminates such a transcoding element, even if it is followed by more combining characters (which may combine with the previous characters for rendering, but which do not affect transcoding). A transcoding text element can include at most one variant hint.

Example 6: Examples 3b and 4b adjusted to use a transcoding variant hint (0xNN00)

3b') □ — Big 5 0xA14E ⟷ Unicode 0x3001+0xNN00

4b') 2 — Mac OS Korean 0xA442 ⟷ Unicode 0x0032+0x20DE+0xNN00

A transcoding variant hint has no effect on any process except transcoding. In particular, variant hints have no effect on the display of Unicode text that contains them. A particular variant hint character does not always imply a particular type of variant (e.g. duplicate, vertical form, bold); the variant hints are *not* intended to carry inline style information.

However, it *is* useful to have consistency in the way a particular variant hint is used with a particular Unicode character. If when mapping Big 5 to Unicode I use a particular variant hint to mean "horizontal form" when combined with Unicode 0x3001, then it is useful if you make the same assignment when mapping EACC to Unicode.

To facilitate this, I propose a simple online registry. For any Unicode character, this could list the

way that particular variant hints have been used with that character, as in the following example. Anyone who needs to use a variant hint with a particular Unicode character could check to see if a similar hint is already in the registry; if not, they could add one.

Example 7: Sample registry information for transcoding variant hints

Unicode 0x3001
+ variant hint 0xNN00: maps to horizontal form (Big 5, EACC)
+ variant hint 0xNN01: maps to bold version (...)
Unicode 0x3002
+ variant hint 0xNN00: maps to large version (...)
+ variant hint 0xNN01: maps to duplicate (...)

The set of transcoding variant hints is small, since it is only intended to be used for variants that exist as separate code points in existing encodings but as single code points in Unicode.

2. Add a set of 4 "transcoding grouping hint" characters (0xNN10-0xNN13)

A "transcoding grouping hint" is used in a manner analogous to the characters in the combining Han proposal. It precedes a specific number of Unicode characters (which may be base characters or combining characters) which are to be treated as a single element for transcoding.

The proposed set of characters is as follows:

- 0xNN10: group next 2 characters for transcoding
- 0xNN11: group next 3 characters for transcoding
- 0xNN12: group next 4 characters for transcoding
- 0xNN13: group next 5 characters for transcoding

Example:

Example 8: Example 5b adjusted to use a transcoding grouping hint (0xNN11)

		Mac OS Japanese		Unicode
5b')	XIV	0x85AC	⟶	0xNN11+0x0058+0x0049+0x0056

The group of characters may begin with a variant hint. This permits roundtrip mapping when the legacy encoding contains multiple variants of characters that must be represented in Unicode using grouping hints. The legacy encoding might contain both regular and bold versions of parenthesized Latin capital letters as single characters, for example.

In this basic proposal, transcoding grouping hints—like variant hints—have no effect on any process except transcoding; they do not affect display. However, see section 4 below for a possible extension to the grouping hint mechanism that would affect display.

3. General comments on transcoding hint characters

A transcoding hint can be discarded or ignored with no loss of basic text content. This is the fundamental point about using transcoding hint characters.

Adding these characters to the BMP provides a mechanism that can relieve the pressure to add

many new Unicode characters for backward compatibility and roundtrip fidelity. In fact, had this mechanism been added earlier, we could have avoided adding many of the existing compatibility characters to Unicode.

The transcoding hints as currently proposed do not require any completely new types of scanning mechanisms for Unicode text. The variant hints can be processed like combining characters, and the grouping hints can be processed like the ideographic structure characters in the combining Han proposal.

At Apple we have been using similar transcoding hint characters (assigned in the private-use area) with our mapping tables for one to two years (depending on the table), so we have some experience with this mechanism. See the Mac OS mapping tables for explanations and examples of use.

4. Possible extensions to grouping hint mechanism

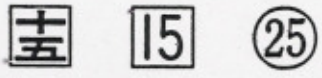
As proposed above, the transcoding grouping hints have no effect on the display of Unicode text that contains them. However, this particular mechanism would provide some interesting advantages if it were extended to affect Unicode text rendering.

First, the grouping hints could specify the direction class information that applies to the entire group; this maintains the existing paradigm that the first character in a sequence (as with a base character followed by non-spacing marks) indicates the attributes of the entire sequence. Consider the single legacy character for Roman number fourteen mapped into a Unicode sequence, as in Example 8 above. The single Unicode characters for Roman numerals have direction class "neutral." However, in example 8 the single legacy character is mapped to a grouping hint followed by a sequence of strong left-right characters; it would be nice if the grouping hint could specify a direction class of neutral for the group. To do this we would need a slightly larger set of grouping hints—say 12, to indicate the direction classes neutral, strong left-right, and strong right-left.

Second, the grouping hints could be extended to allow a non-spacing mark or other combining mark to apply to a *sequence* of Unicode characters, instead of to a single character. The grouping hint would bind more tightly than the combining mark, so a sequence specified by a grouping hint could be followed by a combining mark which would then apply to the sequence as a whole. This would eliminate the need for combining double diacritic characters such as U+0360 and U+0361. It would also provide ways to represent the following using a sequence of basic Unicode characters (instead of assigning separate, single Unicode characters):

Example 9: Benefits of extending the grouping hint mechanism

Some legacy characters (from Mac OS Korean) that could be represented as a grouped sequence of standard Unicode characters plus a combining enclosing mark):



C. Questions & Answers

1. Why not use private-use characters for this purpose?

There are several advantages to assigning these out of the current private use area:

- A general Unicode text consumer would know to completely ignore them for display, text processing, etc. This is not usually the case with private-use characters.

- The properties of the characters would be well-defined.
- They would not impinge on existing usage of private use characters.
- They could help avoid further compatibility additions to Unicode.

2. How does this proposal relate to the Plane 14 variant tag proposal by Hideki Hiura and Tatsuo Kobayashi: L2/97-260 (Dec. 1, 1997)?

Document L2/97-260 proposes tags to indicate display variants (primarily for Han characters), whereas the transcoding hints proposed here explicitly do not affect display.

Document L2/97-260 proposes a set of 256 tags for display variants. The Han character example in that document with the largest number of variants has 35 variants (although those variants really apply to two different Unicode characters, so the number of variants per Unicode character is less than 32). For transcoding purposes, a much smaller set of variant hints will suffice.

Document L2/97-260 proposes adding tags in Plane 14. However, the transcoding hints should be added in the BMP to provide a mechanism that can avoid future compatibility additions to the BMP.

D. Complete example (Big 5)

As an example, here are the current problems with the UTC mapping table for Big 5, and how we could solve them with transcoding hints. In that table, Glenn Adams and John Jenkins note that it is "currently impossible to provide round-trip compatibility between BIG5 and Unicode." Some of the Big 5 characters have no mappings at all in the UTC table, and some have rather incorrect mappings.

Big 5 characters with no UTC mappings at all:

Big 5	Description	Problem	New mapping
0xA15A	Spacing underscore	Duplicates 0xA1C4 (which maps to U+FF3F, FULLWIDTH LOW LINE)	U+FF3F plus hint
0xA1C3	Spacing heavy overscore	No Unicode equivalent	U+203E (OVERLINE) plus hint
0xA1C5	Spacing heavy underscore	No Unicode equivalent	U+FF3F (FULLWIDTH LOW LINE) plus hint
0xA1FE	Diagonal line upper right to lower left	Duplicates 0xA2AC (which maps to U+2571, BOX DRAWINGS LIGHT DIAGONAL UPPER RIGHT TO LOWER LEFT)	U+FF0F (FULLWIDTH SOLIDUS) plus hint
0xA240	Diagonal line upper left to lower right	Duplicates 0xA2AD (which maps to U+2572, BOX DRAWINGS LIGHT DIAGONAL UPPER LEFT TO LOWER RIGHT)	U+FF3C (FULLWIDTH REVERSE SOLIDUS) plus hint
0xA2CC	Hangzhou numeral ten	Duplicates 0xA451 (which maps to Unihan U+5341)	U+5341 plus hint
0xA2CE	Hangzhou numeral thirty	Duplicates 0xA4CA (which maps to Unihan U+5345)	U+5345 plus hint

Big 5 characters which the UTC table maps to incorrect Unicodes to avoid conflicts:

Big 5	Description	Problem	New mapping
0xA14D-0xA154	Alternate punctuation forms for horizontal text or for PRC-style vertical text (different period position than vertical text in Taiwan)	No Unicode equivalent; UTC table maps these to small or halfwidth forms (e.g. U+FE50, U+FF64)	Fullwidth or CJK punctuation (e.g. U+FF0C, U+3001) plus hint
0xA17D-0xA1A4	Alternate (centered) forms for paired punctuation	No Unicode equivalent; UTC table maps these to small forms (e.g. U+FE59, U+FE5D)	Fullwidth or CJK punctuation (e.g. U+FF08, U+3014) plus hint
0xA1CB	Bolder version of 0xA1CA, wavy overline	No Unicode equivalent; UTC table maps this to U+FE4C, DOUBLE WAVY OVERLINE	U+FE4B (WAVY OVERLINE) plus hint
0xA279	Centered vertical line	Duplicate of 0xA278 (which maps to U+2502, BOX DRAWINGS LIGHT VERTICAL); UTC table maps this to U+2595, RIGHT ONE EIGHTH BLOCK, even though it is not on the right	U+2502 plus hint