L2/99-098

US ballot comments on 2nd FCD 14651 International String Ordering

March 16, 1999

The US votes NO on 14651.
Its vote would be changed to YES if the following changes were made.

The main goals of the UTC and US position are to ensure that

(1) Major collation implementations (POSIX, Java, Sybase, etc.) that currently produce satisfactory international orderings for Unicode can be conformant to ISO 14651, and

(2) The proposed Unicode Standard Collation Algorithm (UCA), which pays close attention to the special requirements of Unicode conformance, can be conformant to 14651. The specification of the UCA can be found at http://www.unicode.org/unicode/reports/tr10/.

TECHNICAL comments

The main changes that the UTC requires of 14651 can be summarized as:

A. Levels

Conformant 14651 implementations must not be required to support more than the first 3 levels. (They are free to support more than 3, but not required to.) It is not at all clear from the current conformance clause how many levels a conformant implementation must support. To address this concern, make the following changes:

a. On page 5, 6.2.1.1 Assumptions. The statement that "The number of levels can be extended in the tailoring phase by the end-user." should be modified to: "The number of levels can be extended or reduced in the tailoring phase." (Note also removal of the red-herring use of the term "end-user".)

b. Add the following language to 6.2.1.1
"Conformant implementations of 14651 must support at least three levels. They may support more levels, but they are not required to for conformance. In the absence of such support, fourth and higher level information can be ignored."

B. Position

Conformant 14651 implementations must not be required to support the position designator. (They are free to support the position designator, but not required to.) In addition, the text following the paragraph in 6.2.2.2 starting with "Generally" is informative, not normative, and does not belong in this section.

To address these requirements, make the following changes:

On page 5, 6.2.1.1 Assumptions. The sentence starting "The user shall take care that,..." should be omitted. It is very strange in that it normatively requires a user to "take care that...", but what they must take care is then expressed as a conditional with a protasis expressed as "so that the last level may processed [sic]". The whole sentence is an incomprehensible admonition as it stands. What we want is a clear statement that the standard does not *require* special processing at the last level, but does *allow* it (see below).

In 6.2.1.2, change "A specific property" to "An optional property"

In the first paragraph of 6.2.2.2, change the condition to read:
"If there is an order_start entry that does not use the position value at level m of a block, or if there is no order_start entry, then the formation of subkey level m is done in exactly the same way as the above-defined formation.
Otherwise..."

Add the following language to 6.2.2.2 after the paragraph starting "During".

"Conformant implementations of 14651 are not required to support the position value. They may support this value, but are not required to for conformance. In the absence of such support, the position value is ignored."

d. Split 6.2.2.2 into two parts. The new part 6.2.2.3 would begin on the bottom of page 6, just above the paragraph starting "Generally," and should be entitled: "General interpretation of each level in the Common Template Table".

e. In the new 6.2.2.3, delete all but the first sentence in the paragraph labeled "Level 4". That would disconnect the interpretation of Level 4 from whether or not keys are constructed for Level 4 using the position mechanism.

f. Move the paragraph following the "Level 4" paragraph (starting "In the table, this behavior is...") up into 6.2.2.2 after the note about forward and backward scanning.

g. Move the new section 6.2.2.3 into some other place in the standard. It is informative, and should not be part of the normative clause 6.

C. Backward

Conformant 14651 implementations must not be required to support the backward designator at any level but level 2. Moreover, conformant 14651 implementations are not required to have anything but a **global** backwards switch (e.g. that all weights at a particular level are either uniformly forward or backward). (They are free to support the multiple levels of backwards, and fine-grained directionality [on a per character basis], but not required to.) To address this requirement, add the following language to 6.2.1.2:

"Conformant implementations of 14651 are not required to support the 'backward' scanning direction at any level but level 2. In the absence of such support, the scanning direction is treated as if it were 'forward' at every level but level 2.

"Conformant implementations of 14651 are also not required to support different scanning directions for different blocks. In the absence of such support, if any block has a backward

scanning direction for any level, then all blocks are considered to have that scanning direction at that level."

To the note at the end of 6.2.1.2 starting "In ISO/IEC 10646-1, Arabic…, add the following text: "However, the Unicode Standard does proscribe the logical order of all characters, including Arabic and Hebrew. Implementations conforming to the Unicode standard will not use the backward scanning property."

[Note: the current description of per-block backward and forwards support in 14651 does not serve the goal it was designed for. Since languages and scripts share a great many characters in common, a choice of either forward or backward will cause those common characters to disrupt the order within text of the other direction. For example, suppose Greek is ordered forwards, and French backwards. If digits, for example, are forward then they disrupt the French accents. If they are backward, then they will disrupt the Greek accents.

Even going to a forward, backward, neutral model, as in UCA Version 2 will not work. No matter which heuristics are used to assign the direction of the neutrals, sometimes the choice will be incorrect.

Mixing blocks of different direction is not well supported in industry practice. Most implementations of POSIX do not support it, nor does Java. Forcing these implementations to revise without solid justification is unwarranted. However, as long as implementations are not forced to implement mixed scanning directions, the current language can remain.]

D. Unicode conformance

ISO 14651 must permit a conformant implementation to do the following. (These are required for conformance to the Unicode Standard.)

D.1. Treat canonical equivalent strings as precisely equal in ordering.
D.2. Perform Thai/Lao-style character reversal (see UCA Step 1).
D.3. Exclude irrelevant combining marks when looking up matches for contracting characters (see UCA Step 2).
D.4. Exclude unsupported characters from a collation ordering, or cause them to be sorted in Unicode code point order.

Items D.1 through D.3 are probably covered by section 6.1. However, to ensure that they are, these three items must be added in Notes as examples of conformant implementations, with the following language:

"Note: to allow conformance to the Unicode Standard, conformant implementations may
a. Treat canonical equivalent strings as precisely equal in ordering.
b. Perform Thai/Lao-style character reversal.
c. Exclude irrelevant combining marks when looking up matches for contracting characters.
For more information, see Unicode Technical Report #10."

D.4 is commonly implemented as UNDEFINED in POSIX and other standards. It must be included so that implementations working in low-memory environments that do not need the full default collation rules can use a small subset, and have all other Unicode characters sorted by code order. To fix this problem, make the following changes:

In 6.3.1 rule 23, add the text " | UNDEFINED" to the end of the line.

At the end of 6.2.2.1, add the text:

"If there are no tokens corresponding to a character of the input string, then the character is undefined. Undefined characters are sorted with respect to defined characters as if they were at the position UNDEFINED in the Template Table. (If there is no UNDEFINED token in the table, then the table is interpreted as if there were one at the very end.) The ordering of undefined characters with respect to other undefined characters is not specified by this standard.

Note: there are two common treatments of UNDEFINED characters. The first is to sort among them as if their level-one wieght differences were based upon their UCS character code. The second is to sort them as if they all had the same level-one weight, and their second-level weights were the same as their UCS character codes."

E. Stability:

The data for both UCA and 14651 must be updated to the level of symdump-2.1.9.txt on the SC22/WG20 server (incorporating all of the individual changes that the US would be asking for).

No further changes to other parts of 14651 that would substantially affect the current major collation implementations are acceptable to the UTC or the US national body. In particular, the default data for levels 1, 2, and 3 used by 14651 must be consistent with the UCA data (though perhaps not in the same format). The data was synchronized; this must not diverge due to ballot comments.

F. Specific Technical Comments

Section 6.3.3. is not well defined. Rule I2 (reorder_after) must state what the relationship is between the table lines (X) between the entries and the tailored line containing the symbol defintion (S). That is, suppose we have the following rules:

<UA> <A1>;<A2>;<A3>;<A4>
<UB> <B1>;<B2>;<B3>;<B4>
…
<UX> <X1>;<X2>;<X3>;<X4>
<UY> <Y1>;<Y2>;<Y3>;<Y4>

We want to tailor that table by adding a reordering rule:

reorder-after <UX>
<UX> <X1>;<X2>;<X3>;<X4>
<UY> <Y1>;<Y2>;<Y3>;<Y4>
reorder-end

What does the normalized output (I4) look like? According to the rules, it could be:

<UA> <A1>;<A2>;<A3>;<A4>
<UX> <A1+1>;<MIN2>;<MIN3>;<MIN4>

<UY> <Y1>;<Y2>;<Y3>;<Y4>
<UB> …

Or it could be

<UA> <A1>;<A2>;<A3>;<A4>
<UX> <A1>;<A2>;<A3>;<A4>+1
<UY> <Y1>;<Y2>;<Y3>;<Y4>
<UB> …

Both of these operations might be required for a tailoring, but the rules I1 and I2 do not distinguish between them. Moreover, the rules do not say what is the effect on UB--does it have the same level distinction with the last of the new line(s) that it used to with UA?

To address this problem, the following (or equivalent) change must be made.

6.3.1, rule 32. Change to:
reorder_after_entry := 'reorder-after ' target_symbol ' at level ' digit+

6.3.3 rule I2. Add:
" The reorder entry effectively inserts lines X through Y between existing lines A and B, producing the new ordering <A, X…Y, B>. The level of the reorder-after statement determines the level of the differences between A and X. The level of the difference between Y and B is the stronger of the old difference level between A and B and the new difference level between A and X. For example, suppose we have the following lines (where B1 != A1):

<UA> <A1>;<A2>;<A3>;<A4>
<UB> <B1>;<B2>;<B3>;<B4>
…
reorder-after <UX> at level 2
<UX> <X1>;<X2>;<X3>;<X4>
<UY> <Y1>;<Y2>;<Y3>;<Y4>
reorder-end

will produce the normalized result equivalent to:

<UA> <A1>;<A2>;<A3>;<A4>
<UX> <A1>;<A2>+2;<MIN3>;<MIN4>
<UY> <Y1>;<Y2>;<Y3>;<Y4>
<UB> <Y1>+1; <MIN2>;<MIN3>;<MIN4>"

It must be clearly stated that a reorder-entry also *removes* the lines from where they used to be.

In addition, the following text must be added at the end.
"The reorder-entries must be processed in order during normalization, otherwise incorrect results will be obtained."

The rule I3 also unclear in that it doesn't discuss changing the actual numerical values of the weights. Yet the assignment of numerical values to weights doesn't occur until I5. If the assignment is not done in the reordering, then the subsequent assignment of weights would defeat the purpose of the reordering. This must be clarified.

G. Unicode Reference

Given their importance in the development of this standard, and the fact that the vast majority of 10646 implementations are in fact Unicode implementations, the Unicode Standard must be referenced in Section 3, and Unicode 2.0, TR #8, and DTR #10 must be referenced in the Bibliography.

EDITORIAL Comments:

A. The BNF rules in 6.3.1 should be supplemented by a textual description of the format. The well-formedness conditions can be interleaved with the textual description for clarity.

B. Examples must be added to 6.3.3 to make the requirements clear, as above.

C. Change the explanation in 6.3.1 BNF Syntax Rules to fully describe the notation (e.g. Aho and Ullman):

"<…>	refers to terms not defined in this BNF syntax, and assume general English usage.
'…'	refers to literal characters
(…)	used for grouping
X Y	matches the token sequence X followed by Y
X | Y	matches either X or Y tokens
X*	matches zero or more repetitions of X
X+	matches one or more repetitions of X
{X}	matches one or more repetitions of X "

D. Certain wordsmithing needs to be done for clarity and accuracy. Take the introduction alone:

- Sentence #2 is untrue--that is not the only purpose; others are mentioned below.
- #4 is has an incorrect reference "English" is not a "past approach".
- The last sentence of para#2 is incorrect--one does not "achieve challenges"; one might "overcome them", if that is what is meant.
- "result discrepancies" must be changed to "discrepancies in results"
- "excellent" sounds like blowing our own horn too much.

A full list would take too long to compile -- marked-up copies will be brought to the Pennsylvania meeting.

Introduction, page iv, first paragraph

a) The meaning of the word "**universal**" is ambiguous here. It perhaps implies that there may be other non-Universal properties which are not retained during tailoring. Does this paragraph intend to indicate that all scripts have these properties, or does it mean that the particular values of these properties as defined for each script is common to all users of the Common Template Table, if they are not tailored? One can presume the latter, but it should be more clearly stated. A suggestion might be to change "retaining universal properties for other scripts" to "retaining properties already defined for other scripts."

b) This paragraph seems to be saying that the purpose of this standard is to improve on collation algorithms based only on binary coded character values. If this refers to the use of the binary

coded values without associating a weight to those values, then the next comment about English, with uppercase characters only and no punctuation, being an exception, makes sense. However, it is a rather weak statement, given that even the simplest collation algorithms generally apply some weighting scheme. A suggestion might be to simply delete the remainder of the paragraph beginning with “The purpose of such a mechanism…”

Introduction, page iv, second paragraph

In the first sentence “this is one of the major flaws that affect portability…” it is not clear what “this” is referring to, or what is “flawed”. A suggestion might be to combine the sentence with the parenthetical remark: “That different programs use different ordering specifications is a significant problem reducing portability between countries and between applications.”

Section 1 Scope

In the first paragraph “A simple method of reference…” delete “of reference”, as the method is for comparing not for referencing. It is understood that this standard is defining a method which can be a reference for international ordering.

In the last bullet in this section, delete the final 2 words “to order” in “A context-dependent ordering which would require complex transformation of data to order.”

Section 2 Conformance

The requirements imposed by the second paragraph are unclear.
In the last sentence “and how the comparison method they use If different” the “I” in “if” should not be capitalized. There should be a comma after the word “use”.

Section 4 Definitions

4.6 delta- change “relatively” to “relative”

4.8 graphic character- change
“To a graphic character normally corresponds a glyph.” to
“A graphic character normally corresponds to a glyph.”

4.9 level- This definition is ambiguous as “depth” is not defined. The author should provide a more meaningful definition.

The word "token" should be replaced throughout the document by "weight", unless the definition is in error.
Collating symbol and collating element should be change to collation symbol and collation element.
The difference between ordering key and collation element is not clear from the definitions.
"preparation": speaking of the actual source strings being modified here and in 6.1.1 is worrysome--it is copies of the source strings that are modified, if anything.

Section 5 Symbols and abbreviations

The last 2 sentences in the first paragraph can be worded more grammatically correct and “covered” can be clarified by changing
“What is being referenced is a graphic character, independently of its coding, and any character set whose subrepertoire is taken into account in ISO/IEC 10646-1 is covered in this way.” to
“This is a way to reference a graphic character, independent of its coding. Any character set whose subrepertoire is taken into account in ISO/IEC 10646-1, is included in this specification by this
nomenclature.”

Section 6.1.1 Preparation of character strings prior to comparison

In the first paragraph, will the reference to telephone-book ordering be universally understood, or should the specific problem referred to in this example be brought out?

In the second paragraph, the words “but not both” should be added to the phrase “An application conformant to this internaitonal standard shall at the minimum prepare the string so that sequences using either combining sequences or using precomposed characters…”

In Note 1 of this section, remove the extraneous “ a ” in “precomposed characters affected by a diacritics,”

The term “double-coding” may be unclear. The last sentence might be restated as follows for clarity:

Section 6.2.1.1 Matrix of n lines …

6.2.1.1 "matrix of n lines. N is the number of characters in the repertoire used."
This would exclude multiple characters sorting as 1. Also, "matrix" is unclear; what is meant? It is also not really a "transformation table". What it is is a mapping table from character sequences to collation elements.

Section 6.2.2 Key composition and

Section 6.2.2.1 Formation of subey level 1 through (m-1)

This section is very unclear and must be made more precise and would greatly benefit from an example. In particular, references to directionality are made with respect to string processing, levels and characters and is hard to understand. Stacking is described but unstacking is left to the reader’s imagination. In particular it is not clear when to unstack.

For example, in the second paragraph after the parenthetical remark, it states: “and the new direction is backward” it is not clear how many attributes of the algorithm are affected. The character has the property of being backward, this changes the direction of the current level i, and might be presumed to also affect the scanning direction of the input character string, which is described as initially forward in the first paragraph.

If we understand the proposed algorithm correctly, it would benefit the specification to state clearly:

1) That scanning of the input character string is always forward thru the logical sequence of the string.
2) That reaching a character with a backwards property changes the current direction of level i from forward to backward, and commences stacking of position and token.
3) That reaching a character with a forwards property when the current direction of level i is backwards, changes the level's direction to forwards and commences unstacking, with a description of what is involved in unstacking.

Section 6.2.2.2 Formation of subkey level m

The first sentence should change "uses" to "use".

The first paragraph begins with discussion of **order_start_entry** which is not yet introduced . This should be characterized and the subsequent reference to having or not having a position, expanded upon for clarity.
The significance of using the table as-is versus changing it in accordance with frequent market practice should also be clarified and the alternative behaviors of the ordering described. An explanation of why the Common Template Table does not follow frequent market practice might also be offered.

In the second paragraph, the sentence "When the character is not assigned at level m in the table, it is ignored for the formation of subkey level m and no pair is concatenated." Might be better moved to the end of the paragraph, so the subsequent sentences cannot be perceived to be part of the condition "when the character is not assigned at level m".

In addition, this paragraph is the first indication that a character might not have entries for every table level. There should be some discussion of this and its impact on behavior of the ordering.

The first sentence in the description of level 4 states: "This level represents the level common to all scripts or the level not specifically belonging to any script." We do not understand what this means. How and why is this level different from the other levels?

In the last paragraph of this section, it is stated: "In the Common Template table, definitions of these characters for level 1 to 3…". We do not understand which characters are referred to by "these characters".
Perhaps the author should state: "In the Common Template table, characters that are assigned values at level 4, are exclusively assigned to level 4, and are ignorable, and have no values assigned, at levels 1-3.

It might improve the readability and understandability of the specification, if the actual description of the Common Template table was moved out of this section to the later section on the Common Template table and if the information in level 4, about the formation of the level 4 or level m subkey, was included with the first 2 paragraphs of this section, describing the key formation.

Section 6.3.4. :

The first paragraph can be simplified considerably to:
Two collation weighting tables are said to be equivalent if any comparison of strings using those tables results in the same ordering.

Section 6.4 Declaration of a delta

In the second paragraph, conformance is described as declarable if a fixed table is used by the application. Can an application conform if it does not make use of a fixed table analagous to the Common Template table?

Also, the term “comparison table” is not defined. Presumably this is the name for the transformation table used with the comparison method and this should be stated or clarified. Also the word “relatively” should be “relative” in this instance.

In the first bullet, there is a reference to direction values being dependent on writing systems. Earlier, the specification pointed out that scanning direction is in fact independent of the direction of writing, so this may be confusing and misleading to readers.

In the first paragraph after the 4 bullets, the sentence beginning with “In cases where the applications has…” should be changed to “In cases where the applications have…”.

end of document