ISO/IEC JTC1/SC2/**WG2 N4309**
ISO/IEC JTC1/SC2/WG2/**IRG N1879**
L2/12-241

Universal Multiple-Octet Coded Character Set
International Organization for Standardization

Doc Type: Working Group Document
Title: Response to WG2 N4247 (*Preserving round-trip integrity of CJK compatibility ideographs*)
Author: Mark Davis & Ken Lunde
Source: The Unicode Consortium
Status: Liaison Contribution
Action: For consideration by JTC1/SC2/WG2
Date: 2012-08-03

This document represents the UTC's official response to Masahiro Sekiguchi's WG2 N4247 (*Preserving round-trip integrity of CJK compatibility ideographs*), which is itself a reply to WG2 N4246 (*Stabilizing CJK Compatibility Ideographs through the use of Standardized Variants*). This document is intended to also offer some clarity on the issues that are raised in both documents.

The author of WG2 N4247 first wrote the following:

> The author of N4246 just writes "*it is not possible to guarantee that normalization will not be applied, except for completely closed environments.*" It may be true, but we today have a lot of programs that, say, blindly consider the data is pure ASCII, Shift-JIS, or something else, breaking Unicode data. It is not possible to guarantee that bad handling of data based on wrong encoding assumptions will not be applied, except for completely closed environments. Both of them are sad facts. I'm not sure why the author of N4246 emphasizes the former.

The point of that particular WG2 N4246 statement is simply to draw attention to the fact that normalization cannot be prevented, meaning that it is much safer to assume that normalization will be applied, and to come up with a solution that works for this specific case. A representation that is immune to the effects of normalization is an obvious requirement for a workable solution.

The author then wrote the following:

> **My objection to the recommendation in N4246**
>
> N4246 says "1,002 Standardized Variants ... would be equivalent to the CJK Compatibility Ideographs". It is not clear what the word *equivalent* means in this context. I want clarification how those Standardized Variants are meant to be used.
>
> If the word "equivalent" means "to be normalized", the recommendation can't be a solution to the normalization problem, so I think it means something else.
>
> I have a feeling it means "should be used in place of". If I'm correct, it requires a large amount of updates to the existing applications that depends on compatibility ideographs today, and I don't think the transition is feasible. I don't understand why use think " a wide variety of products, protocols, and environments normalize text data on a regular basis, and this cannot be changed" while use think transition from use of compatibility ideographs to the newly proposed sequences is easy.

The word *equivalent* simply refers to an *alternate representation* that happens to be more stable because it is immune to the effects of normalization.

Note that nothing in WG2 N4246 would prevent the continued use of CJK Compatibility Ideographs, but those who choose to do so simply need to be aware of the ramifications, such as the effects of normalization.

In terms of the transition that is suggested by WG2 N4246, keep in mind that CJK Compatibility Ideographs are effectively broken, because the application of any of the four normalization forms will cause them to lose their dis-

tinctions from their canonical equivalents, all of which are CJK Unified Ideographs. Transitioning from a broken to unbroken state requires a transition. Put simply, the sooner that a solution is standardized, the sooner that CJK Compatibility Ideographs can be fixed.

The following is the author's first suggested alternative to the solution that was proposed in WG2 N4246:

> Alternative 1
>
> The first alternative proposed here is to define several other types of normalization forms that preserve compatibility ideographs as they are while normalizing composite sequences and precomposed characters. We need further study before deciding which parts of the mappings should be kept and which parts should be changed in details, but I believe it's doable.
>
> The best part of this alternative is that most of the transcoding-dependent applications require very small changes; when the new normalization forms are supported on the platforms, application programs simply invoke the normalization services to use the newly-defined normalization. It is a big difference from the N4246 recommendation; it requires rewriting of all existing application that use compatibility ideographs to use the newly introduced *standardized variant sequences*.

The four normalization forms defined by the Unicode Standard are in extremely widespread use. Adding additional forms of normalization is impractical at this point. Moreover, new forms of normalization differing only by the handling of CJK Compatibility Ideographs would be almost identical to the current forms. The addition of such new forms would simply muddy the waters: in practical terms, they would cause the situation to become even worse. As discussed below, all it would take to cause a failure in a chain of processes would be for a single process in that chain to apply the original normalization form on the text data.

The following is the author's second suggested alternative::

> Alternative 2
>
> Recommend everyone to keep all Unicode data in their original no-normalized forms, and perform normalization only before comparison or similar operation, discarding the normalized data soon.
>
> I understand this changes today's practice. I also remember the discussion took place in a W3C working group that published a "normalize everything before send it to the Net" recommendation. However, there are good chances this change is accepted by the community, because security environment surrounding the Internet application changed in the past decade, and application that communicate through Internet anyway need to normalize all received data before any normalization-critical operations since it can't assume the received data from someone else is properly normalized.
>
> The transition need not be immediate nor synchronized. Yes, it will require some long time to this transition to complete all over the world, but anybody who concerns problem can start early without waiting others. This seems a big plus of this alternative.

There are now countless processes that interact with each other, and these processes can act on each other's text data. All it takes is a single process to apply normalization on the text data, and the result is that the distinctions that were intended to be preserved by the CJK Compatibility Ideographs are instantly lost. Statistically, failure would be almost certain because of the distributed nature of processing, asynchronous rollout of software updates, and so on.

The inherent problem with this alternative is that it is not possible to contact all developers, and that even if they could be contacted, to guarantee that they will change the processes over which they have control. Effectively, this is the same kind of problem as would be involved in trying to change the underlying architecture of DNS.

The Standardized Variants solution that is proposed in WG2 N4246 is attractive in that CJK Compatibility Ideographs can continue to be used, but that developers who have customers that are effected by this problem can convert their text data to use this solution, thus preserving their distinctions. As opposed to the problems of new normalization forms, the Standardized Variants are robust; once the characters are converted to that form, they survive Unicode normalization by any process in a chain of processes. We predict that, over time, the Standardized Variants will become the preferred representation for CJK Compatibility Ideographs.

> What is the problem, by the way?
>
> The document N4246 is busy to explain what happens when normalization is applied to compatibility ideographs and *solution*, it is silent on what is the problem. As an engineer working for an IT vendor, I receive many tough issues regarding Unicode or other codeset issues though

> the company's sales and support division in regular basis, but I have never heard of complains something like "My compatibility ideograph was lost after normalization!" I doubt you are only discussing possibilities of problems but real problems.
>
> On the other hand, I sometimes receive query on composite sequence and precomposed character comparison issues recently, especially those regarding voiced kana letters. I sometimes recommend normalize-before-compare strategy, but, of course, I can't always because it breaks round-trip integrity if the application requires one. My problem is hence: how I can process compatibility ideographs maintaining round-trip integrity, while recognizing a corresponding composite sequence and a precomposed character identical. I have no easy and universal solution to the problem.
>
> I love to work with those who share the same concern for a good solution (not limited to two proposals stated in this document,) although it is not appropriate to say it here, since it seems a separate goal from N4246.

The problem is very real, and large companies with enormous global customer bases, such as Adobe, Apple, Google, IBM, Microsoft, and others, face this issue—normalization of CJK Compatibility Ideographs—on a regular and on-going basis. Their OSes, applications, and text engines routinely interact with processes that are developed by other companies.

Among the 1,002 CJK Compatibility Ideographs, the ones that are affected most by this problem are those that are used in Japan, specifically the ones that correspond to kanji in the JIS X 0213:2004 standard. The solution proposed in WG2 N4246 is intended to handle all 1,002 CJK Compatibility Ideographs.