L2/02-396

Title: **In defense of Plane 14 language tags**
Source: **Doug Ewell**
Status: **Individual contribution**
Action: **For consideration by UTC**
Date: **2002-11-01**

On October 23, 2002, the Unicode Consortium published a list of "Open Issues for Public Review" which included (as Issue #1) a proposal to "Deprecate the Plane 14 Language Tags" introduced in Unicode Technical Report #7 and subsequently added to Unicode 3.1.

The public-review process had been proposed at the UTC 91/L2 188 joint meeting on May 2, 2002, with the specific goal of reviewing "candidates for deprecation" and in particular the deprecation of Plane 14 language tags.

This paper will attempt to show why the Plane 14 language tag characters should not be deprecated, and in fact should be made available for use beyond the "special protocols" mentioned in Unicode Standard Annex #27. They constitute a simple, extensible mechanism for solving specific problems with minimal overhead.

Readers are asked to consider the following arguments individually, so that any particular argument that seems untenable or contrary to consensus does not affect the validity of other arguments. Also, please bear in mind that since I am not a member of WG2, the UTC, or the Consortium, I may not be familiar with issues surrounding Plane 14 tags that have been discussed only in meetings or on the Unicore list. I will address those issues that I am aware of and have seen discussed on the public Unicode list.

1. Language tags may be useful for display issues.

The most commonly suggested use, and the original impetus, for Plane 14 language tags is to suggest to the display subsystem that "Chinese-style" or "Japanese-style" glyphs are preferred for unified Han characters. (Korean and Vietnamese usages of Han are not usually mentioned in this context.) A frequent rebuttal is that the language of the text does not provide adequate information to determine the preferred glyph style. For example, it is often said that Japanese users prefer "Japanese-style" glyphs universally, even for Chinese text.

The Plane 14 tagging approach is not perfect, but it *is* sufficient to solve this problem. Japanese users who prefer "Japanese-style" glyphs universally can tag all Han text as "ja", which may be linguistically wrong but achieves the desired effect. Users who want Chinese

glyphs for Chinese-language text and Japanese glyphs for Japanese-language text can tag the former as "zh" and the latter as "ja" as they see fit.

Other scripts besides Han can benefit from plain-text language tagging as well. A common Latin-script example is that acute accents over Polish letters have a noticeably steeper slant than they do over (e.g.) French letters. Fonts are not usually designed to put a steeper mark over a letter like *ź* used in Polish and a more horizontal mark over a letter like *á* used in French. Language tags can be used in conjunction with display engines to indicate a preference between alternative glyphs in such cases.

A more ambitious display engine might attempt to render Arabic-script text in *nastaliq* style if the text is tagged "ur" (Urdu).

This application of Plane 14 language tagging is lightweight and extensible, works in all types of text, and is not tied to any particular display technology or vendor, unlike solutions based on proprietary font formats and word processors.

2. Language tags may be useful for non-display issues.

Although not frequently mentioned, plain-text language tagging could also be useful for applications such as speech synthesis, spell-checking, and grammar checking. Uses such as this are mentioned in RFC 3066, "Tags for the Identification of Languages." Again, the Plane 14 approach works with all Unicode text, not just HTML or other rich-text formats.

3. Conflict with HTML/XML tags need not be a problem.

A common criticism of the Plane 14 language tags is that higher-level protocols such as HTML and XML already provide a mechanism for language tagging. There is a concern that the language specified by the "lang" attribute in HTML or "xml:lang" attribute in XML could conflict with the one specified in a Plane 14 language tag, and that the process interpreting this text would be like the proverbial sailor with two watches, not knowing which to follow.

The potential disruption caused by this scenario is probably overstated. Almost every HTML file ever created contains at least one plain-text line separator (`CR` and/or `LF`) *and* at least one HTML-style line separator (`<p>` and/or `<br>`). Which to follow? The HTML specification very clearly states that the higher-level protocol takes precedence in this case (unless `<pre>preformatted text</pre>` is explicitly indicated). The same could be said for the interaction between Plane 14 language tags and HTML language tags. It would be easy to add a passage to Section 8.1 of the HTML 4.01 specification to the effect that Plane 14 tags are to be ignored in HTML, or should be superseded by HTML "lang" tags, without the need to deprecate them in Unicode.

For that matter, Unicode has brought its own potential ambiguity of this type to the table, introducing U+2028 LINE SEPARATOR and U+2029 PARAGRAPH SEPARATOR over 10 years ago. Despite the similar potential for confusion in mixing LS and PS with the much more common CR and LF, I have never seen a proposal to deprecate LS and PS.

4. **The original need for language tags has not disappeared.**

   At one time, so the story goes, a group of CJK users proposed a mechanism for language tagging of plain text that utilized invalid UTF-8 sequences. UTF-8 was still a relatively new format at the time and this proposal would have seriously destabilized it. The Plane 14 language tagging strategy was reportedly devised in response to this proposal, to provide an alternative means of indicating Chinese or Japanese text (= glyphs) while protecting the integrity of UTF-8.

   Plane 14 may not have been picked up by a large number of users to date, but is there any indication that this original need has gone away? Japanese users on the Internet continue to suggest ISO 2022-based solutions (character set switching) as a means of distinguishing Chinese and Japanese text. Deprecating the Plane 14 language tagging mechanism would add fuel to the claim that Unicode "doesn't care" about the Chinese/Japanese glyph issue. Continuing to support Plane 14, and sanctioning its use in situations other than the "standard protocols," could result in greater market acceptance of Unicode in East Asia.

   One Unicode list member, in private e-mail, asked if I would be interested in supporting a "lightweight text XML… with really few tags, maybe only a language tag." If there is *still* a need, or even a perceived need, for language tagging in plain text, does it make more sense to create yet another markup format or to keep what *already exists* in Unicode?

5. **"Statefulness" disadvantage is exaggerated.**

   Plane 14 tags, like all language tagging mechanisms, are stateful. It is sometimes said that statefulness is un-Unicode and a main reason why language tagging belongs to a higher-level protocol. But bidirectionality is also stateful, and so – to a lesser extent – are line breaking and many fine details of rendering (e.g. Arabic contextual forms). It was known at the time UTR #7 was written, and at the time Plane 14 tags were formally added to Unicode 3.1, that they were stateful. Why was this acceptable before and not now?

6. **Plane 14 tags are easy to filter out, and harmless if not interpreted.**

   The argument that Plane 14 tags create an undue hardship on Unicode text processes is simply untrue. They were situated in the empty Plane 14, the last plane not designated for private-use characters, precisely so they would be easy to detect and ignore. Section 13.7 of Unicode 3.1 goes to great lengths to explain how processes can filter out Plane 14 tags, using examples in UTF-8, UTF-16, and UTF-32.

If they are not screened out, Plane 14 language tags are treated just like any other Unicode character – they are ignored. Conformance to the Unicode Standard means that a process that cannot interpret a given character must leave that character alone. Tags are not displayed with a visible glyph, so they do not appear on the user's screen as an inscrutable sequence of black boxes. They do not affect searching, and are unlikely to affect sorting since data items to be sorted would not normally be language-tagged using any mechanism. They do not cause any more trouble for a text processor than an uninterpreted Yi syllable.

7. **Rapid deprecation creates an image of instability.**

   Some individuals in the Internet-protocol and security fields have expressed concerns over perceived instability in the Unicode Standard. Most of these concerns are ill-founded, and demonstrate a lack of understanding of a *living, extensible* standard such as Unicode/10646. Nevertheless, the precedent of adding a new type of functionality to Unicode, only to deprecate it a mere two years later, may create a greater sense of instability among opinion leaders. Creators of Korean implementations for Unicode 1.x may feel a sense of *déjà vu* over the deprecation of Plane 14 language tags.

8. **Other, as yet uninvented tags would be implicitly deprecated.**

   The Plane 14 tagging mechanism was deliberately designed not to be limited to language tags. A single "begin language tag" character was created, along with a mechanism by which the "end tag" character could be used to terminate either a specific tag or all tags. The intent was to allow other types of tags to be created in the future, if a need arose.

   By deprecating the language tags, the UTC runs the risk of implicitly deprecating the entire Plane 14 tagging mechanism, effectively shutting the door on a potential Unicode-based solution to some as-yet unknown problem.

Plane 14 language tags are lightweight, as designed, and impose a negligible burden on users and implementations, especially when compared with other Unicode text processes such as normalization and bidirectionality. They would be more useful if not artificially constrained to unspecified "special protocols." They offer flexibility and cause minimal inconvenience. The proposal to deprecate them should be rejected.