L2/12-286

Subject: What to do about CCC133?
From: Mark Davis
Live: http://goo.gl/fbYUE

Issue

There was a typo in a property value in 6.1

```
ccc; 132; CCC133                 ; CCC133
```

This was clearly wrong, and especially misleading given the relationship between the numeric value and the name. There is no question that we want to introduce CCC132 with the right meaning. The question is what to do about CCC133.

There are basically 3 options.

Option A. Retain deprecated alias

```
ccc; 132; CCC132                 ; CCC132             ; CCC133
```

and (a) permanently reserve 133 to prevent errors, and (b) deprecate the use of CCC133

Option B. Deprecate and remove alias

```
ccc; 132; CCC132                 ; CCC132
```

and (a) permanently reserve 133 to prevent errors, and (b) deprecate the use of CCC133

Option C. Reinterpret alias

```
ccc; 132; CCC132                 ; CCC132
ccc; 133; CCC133                 ; CCC133 // currently empty
```

and (b) document that CCC133 means something different in U6.2 from in U6.2.

This also raises the more important general issue about stability of property aliases and property value aliases.

Details

In normal cases, we do Option A. We have done this before:

```
lb ; IN                     ; Inseparable                  ; Inseperable
```

However, generally people feel like this case is special, that it can cause ongoing confusion into the future, given the close relationship between the numeric value 133 and the name CCC133.

On the other hand, Option C breaks the implicit stability of the property values. Under that proposal:

In Unicode 6.1, [:ccc=132:] is identical with [:ccc=CCC133:]
In Unicode 6.2, [:ccc=132:] is disjoint with [:ccc=CCC133:]

It would be similar in kind to the following hypothetical change to LineBreak.

```
lb ; SA                     ; Complex_Context
=>
lb ; SA                     ; Special_Action
lb ; CC                     ; Complex_Context
```

In Unicode X, [:lb=SA:] is identical, by definition, with [:lb=Complex_Context:]
In Unicode X+1, [:lb=SA:] is disjoint, by definition, with [:lb=Complex_Context:]

That is not what was intended by http://unicode.org/policies/stability_policy.html#Alias_Stability.

“Property aliases and property value aliases, once defined in the Unicode Character Database, will never be removed.”

While the *letter* of that principle would permit changes to CCC133 and Complex_Context above, it would contradict the notes underneath that principle:

- “This stability guarantee makes it possible to use property aliases and property value aliases as stable identifiers.”

Aliases are certainly not stable we can split them up and recombine differently.

What this whole issue points out is that we need need to tighten the language of http://unicode.org/policies/stability_policy.html#Alias_Stability, to something like

S1. Property aliases and property value aliases, once defined in the Unicode Character Database, will never be removed nor assigned to different values.

We could do Option C, but put a “fuzzy” loophole in the stability policies, like:

S1-Note1. “In exceptional cases, retaining a typographical or other mistake may not be feasible, for example because it might be universally mistaken as being an alias for another value. In such cases retention of the mistake for stability would cause more damage over the life of the standard than fixing it, especially if discovered shortly after publication.”

My concern with a policy like this, is that it can introduce a sense of unease in implementers. It can look like if we just don’t like any particular property value in a release, that we can change it afterwards, screwing up their implementations. It makes properties and property values “tentative”.

I think a cleaner approach would be to do Option B, to solve the immediate issue, and add a note like the following:

S1-Note2. “While property aliases and property value aliases cannot be removed, they can be deprecated, and can be changed such that they are “empty”; that no code points have that property or property alias.