

## Comments on L2/18-337 - Broaden the scope of what Unicode calls “properties”

Michael Saboff  
msaboff@apple.com  
January 14, 2019

As *L2/18-337 - Broaden the scope of what Unicode calls “properties”* is currently written, it has as its goal unifying the treatment of sequences with that of properties. It seeks such a clarification from the Unicode TC in order to unblock an ECMA TC-39 proposal.

I serve on ECMA TC-39 and although I support adding the ability to search for Unicode sequences via regular expressions, I oppose reusing property matching syntax for sequences. I believe that such a move would confuse developers and users of the resulting ECMAScript language and imply capabilities that won't exist. I support using new and distinct syntax for sequences.

Furthermore, I believe that what is proposed in *L2/18-337* is incomplete, and it will be detrimental to Unicode and its uses in other ways. I make the following claims in regards to *L2/18-337*.

1. *L2/18-337* does nothing to unify how characters / code-points, and strings are tested for a properties or a sequence. Such testing will still require determining first if what is being tested is a single entity, aka property, or a string, aka sequence, before proceeding. Certainly the current checking for a property could evolve to looping over the elements of a sequence property, but such a looping algorithm would obfuscates and complicates the direct table, multistage table or range lookup needed for most current properties.
2. In most current cases, sequences are boolean, either a string matches or it doesn't. Properties on the other hand can have many possible values. Even the canonical-equivalent sequences are still boolean mappings.
3. In general, sequences are used to describe strings that have meanings distinct from their individual code points. *L2/18-337* is focused primarily on five Emoji sequences. These sequences are lists of strings (or list of sequences) that define a boolean group membership. While *L2/18-337* does provide for future lists of sequences, it doesn't address the myriad of current Unicode sequences, e.g. **LATIN CAPITAL LETTER A WITH OGONEK AND ACUTE**. *L2/18-337* proposes adding **\_Sequence** to the one Emoji sequence that doesn't have that suffix, but does nothing to address all of the existing non-Emoji sequences that don't end in **\_Sequence**. Given the Unicode name stability guarantee, these other sequences would require secondary names or aliases with an added **\_Sequence** suffix.
4. *L2/18-337* is written to support the overloaded use of `\p{<sequence-name>}` in regular expression processing. The TC-39 proposal that would immediately take

advantage of *L2/18-337* limits the overloaded use depending on context. The `\p` escape when used for a sequence would only be used to direct matching of said sequence. The negative `\P{<sequence-name>}` is disallowed and results in an early syntax error. Although `\p{<property-name>}` can be used to build larger character classes (called *character ranges* in UTS #18), the use of `\P{<sequence-name>}` in a character class would also result in an early syntax error, since character classes are a set of code points to match. Both *L2/18-337* and the related TC-39 proposal are built upon the developer understanding that a suffix of `_Sequence` connotes different usage and match processing. It is my contention that given the differing acceptable uses of sequence matching from that of property matching, these differences are best handled with different syntax. The escape syntax of `\q{<sequence-name>}` has been proposed, discussed and is the current alternative to using `\p`. Note that `\s` is already in common use for regular expression syntax to search for whitespace, and therefore the alternate `\q` has suggested. The distinct use of `\p` / `\P` for properties, and `\q` for sequences conveys the difference acceptable usage within regular expressions for each, instead of relying on the entity names for the contained constructs.

5. *L2/18-337* doesn't address the related issues in **UTS #18 Unicode Regular Expressions**. Certainly the use of `\p{<sequence-name>}` outside of a character class would work as expected, but exclusions to using `\P` with sequences should be added to UTS 18 as well as the prohibition of `\p` with sequences inside a range construct. Lastly, establishing a general precedent of overloading `\p` for sequences is best communicated by Unicode instead of a dependent programming language.
6. Finally, by reusing syntax, we will likely introduce incompatibilities in deployed code. Consider existing code that accepts regular expression components as strings, including properties, where a property name is inserted in a `\p{...}` or `\P{...}` construct. Those computed property escapes could run into the syntax issues described above without the associated error handling. Such error handling would likely not be anticipated since it wasn't needed for properties. The only error checking required today is validating if existence of a property, and not its acceptable usage within a regular expression.

I am in favor of adding regular expression support of Unicode Sequences, but through the use of new syntax that clearly communicate the capability and its limitations. Adding sequence support in such a manner scales to the myriad of existing sequences without requiring any sequence renaming or aliasing. I object to blurring the lines between properties and sequences for the aforementioned stated reasons.