                Language Tagging in Unicode Plain Text

Status of this Memo

   This memo provides information for the Internet community. It does
   not specify an Internet standard of any kind. Distribution of this
   memo is unlimited.

Copyright Notice

IESG Note:

   This document has been accepted by ISO/IEC JTC1/SC2/WG2 in meeting
   #34 to be submitted as a recommendation from WG2 for inclusion in
   Plane 14 in part 2 of ISO/IEC 10646.

1. Abstract

   This document proposed a mechanism for language tagging in [UNICODE]
   plain text. A set of special-use tag characters on Plane 14 of
   [ISO10646] (accessible through UTF-8, UTF-16, and UCS-4 encoding
   forms) are proposed for encoding to enable the spelling out of
   ASCII-based string tags using characters which can be strictly
   separated from ordinary text content characters in ISO10646 (or
   UNICODE).

   One tag identification character and one cancel tag character are
   also proposed. In particular, a language tag identification character
   is proposed to identify a language tag string specifically; the
   language tag itself makes use of [RFC1766] language tag strings
   spelled out using the Plane 14 tag characters. Provision of a
   specific, low-overhead mechanism for embedding language tags in plain
   text is aimed at meeting the need of Internet Protocols such as ACAP,
   which require a standard mechanism for marking language in UTF-8
   strings.

   The tagging mechanism as well the characters proposed in this
   document have been approved by the Unicode Consortium for inclusion
   in The Unicode Standard. However, implementation of this decision

awaits formal acceptance by ISO JTC1/SC2/WG2, the working group
responsible for ISO10646. Potential implementers should be aware that
until this formal acceptance occurs, any usage of the characters
proposed herein is strictly experimental and not sanctioned for
standardized character data interchange.

2. Definitions and Notation

No attempt is made to define all terms used in this document. In
particular, the terminology pertaining to the subject of coded
character systems is not explicitly specified. See [UNICODE],
[ISO10646], and [RFC2130] for additional definitions in this area.

2.1 Requirements Notation

This document occasionally uses terms that appear in capital letters.
When the terms "MUST", "SHOULD", "MUST NOT", "SHOULD NOT", and "MAY"
appear capitalized, they are being used to indicate particular
requirements of this specification. A discussion of the meanings of
these terms appears in [RFC2119].

2.2 Definitions

The terms defined below are used in special senses and thus warrant
some clarification.

2.2.1 Tagging

The association of attributes of text with a point or range of the
primary text. (The value of a particular tag is not generally
considered to be a part of the "content" of the text. Typical
examples of tagging is to mark language or font of a portion of
text.)

2.2.2 Annotation

The association of secondary textual content with a point or range of
the primary text. (The value of a particular annotation *is*
considered to be a part of the "content" of the text. Typical
examples include glossing, citations, exemplication, Japanese yomi,
etc.)

2.2.3 Out-of-band

An out-of-band channel conveys a tag in such a way that the textual
content, as encoded, is completely untouched and unmodified. This is
typically done by metadata or hyperstructure of some sort.

2.2.4 In-band

   An in-band channel conveys a tag along with the textual content,
   using the same basic encoding mechanism as the text itself. This is
   done by various means, but an obvious example is SGML markup, where
   the tags are encoded in the same character set as the text and are
   interspersed with and carried along with the text data.

3.0 Background

   There has been much discussion over the last 8 years of language
   tagging and of other kinds of tagging of Unicode plain text. It is
   fair to say that there is more-or-less universal agreement that
   language tagging of Unicode plain text is required for certain
   textual processes. For example, language "hinting" of multilingual
   text is necessary for multilingual spell-checking based on multiple
   dictionaries to work well. Language tagging provides a minimum level
   of required information for text-to-speech processes to work
   correctly. Language tagging is regularly done on web pages, to
   enable selection of alternate content, for example.

   However, there has been a great deal of controversy regarding the
   appropriate placement of language tags. Some have held that the only
   appropriate placement of language tags (or other kinds of tags) is
   out-of-band, making use of attributed text structures or metadata.
   Others have argued that there are requirements for lower-complexity
   in-band mechanisms for language tags (or other tags) in plain text.

   The controversy has been muddied by the existence and widespread use
   of a number of in-band text markup mechanisms (HTML, text/enriched,
   etc.) which enable language tagging, but which imply the use of
   general parsing mechanisms which are deemed too "heavyweight" for
   protocol developers and a number of other applications. The
   difficulty of using general in-band text markup for simple protocols
   derives from the fact that some characters are used both for textual
   content and for the text markup; this makes it more difficult to
   write simple, fast algorithms to find only the textual content and
   ignore the tags, or vice versa. (Think of this as the algorithmic
   equivalent of the difficulty the human reader has attempting to read
   just the content of raw HTML source text without a browser
   interpreting all the markup tags.)

   The Plane 14 proposal addresses the recurrent and persistent call for
   a lighter-weight mechanism for text tagging than typical text markup
   mechanisms in Unicode. It proposes a special set of characters used
   only for tagging. These tag characters can be embedded into plain

text and can be identified and/or ignored with trivial algorithms,
since there is no overloading of usage for these tag characters--they
can only express tag values and never textual content itself.

The Plane 14 proposal is not intended for general annotation of text,
such as textual citations, phonetic readings (e.g. Japanese Yomi),
etc. In its present form, its use is intended to be restriced solely
to specifying in-line language tags. Future extensions may widen
this scope of intended usage.

4.0 Proposal

   This proposal suggests the use of 97 dedicated tag characters encoded
   at the start of Plane 14 of ISO/IEC 10646 consisting of a clone of
   the 94 printable 7-bit ASCII graphic characters and ASCII SPACE, as
   well as a tag identification character and a tag cancel character.

   These tag characters are to be used to spell out any ASCII-based
   tagging scheme which needs to be embedded in Unicode plain text. In
   particular, they can be used to spell out language tags in order to
   meet the expressed requirements of the ACAP protocol and the likely
   requirements of other new protocols following the guidelines of the
   IAB character workshop (RFC 2130).

   The suggested range in Plane 14 for the block reserved for tag
   characters is as follows, expressed in each of the three most
   generally used encoding schemes for ISO/IEC 10646:

   UCS-4

   U-000E0000 .. U-000E007F

   UTF-16

   U+DB40 U+DC00 .. U+DB40 U+DC7F

   UTF-8

   0xF3 0xA0 0x80 0x80 .. 0xF3 0xA0 0x81 0xBF

   Of this range, U-000E0020 .. U-000E007E is the suggested range for
   the ASCII clone tag characters themselves.

4.1 Names for the Tag Characters

   The names for the ASCII clone tag characters should be exactly the
   ISO 10646 names for 7-bit ASCII, prefixed with the word "TAG".

   In addition, there is one tag identification character and a CANCEL
   TAG character. The use and syntax of these characters is described in
   detail below.

   The entire encoding for the proposed Plane 14 tag characters and
   names of those characters can be derived from the following list.
   (The encoded values here and throughout this proposal are listed in
   UCS-4 form, which is easiest to interpret. It is assumed that most
   Unicode applications will, however, be making use either of UTF-16 or
   UTF-8 encoding forms for actual implementation.)

```
   U-000E0000  <reserved>
   U-000E0001  LANGUAGE TAG
   U-000E0002  <reserved>
   U-000E001F  <reserved>
   U-000E0020  TAG SPACE
   U-000E0021  TAG EXCLAMATION MARK
   U-000E0041  TAG LATIN CAPITAL LETTER A
   U-000E007A  TAG LATIN SMALL LETTER Z
   U-000E007E  TAG TILDE
   U-000E007F  CANCEL TAG
```

4.2 Range Checking for Tag Characters

   The range checks required for code testing for tag characters would
   be as follows. The same range check is expressed here in C for each
   of the three significant encoding forms for 10646.

Range check expressed in UCS-4:

```
if ( ( *s >= 0xE0000 ) || ( *s <= 0xE007F ) )
```

Range check expressed in UTF-16 (Unicode):

```
if ( ( *s == 0xDB40 ) && ( *(s+1) >= 0xDC00 ) && ( *(s+1) <= 0xDC7F ) )
```

Expressed in UTF-8:

```
if ( ( *s == 0xF3 ) && ( *(s+1) == 0xA0 ) && ( *(s+2) & 0xE0 == 0x80 )
```

   Because of the choice of the range for the tag characters, it would
   also be possible to express the range check for UCS-4 or UTF-16 in
   terms of bitmask operations, as well.

4.3 Syntax for Embedding Tags

   The use of the Plane 14 tag characters is very simple. In order to
   embed any ASCII-derived tag in Unicode plain text, the tag is simply
   spelled out with the tag characters instead, prefixed with the
   relevant tag identification character. The resultant string is
   embedded directly in the text.

   The tag identification character is used as a mechanism for
   identifying tags of different types. This enables multiple types of
   tags to coexist amicably embedded in plain text and solves the
   problem of delimitation if a tag is concatenated directly onto
   another tag. Although only one type of tag is currently specified,
   namely the language tag, the encoding of other tag identification
   characters in the future would allow for distinct tag types to be
   used.

   No termination character is required for a tag. A tag terminates
   either when the first non Plane 14 Tag Character (i.e. any other
   normal Unicode value) is encountered, or when the next tag
   identification character is encountered.

   All tag arguments must be encoded only with the tag characters U-
   000E0020 .. U-000E007E. No other characters are valid for expressing
   the tag argument.

   A detailed BNF syntax for tags is listed below.

4.4   Tag Scope and Nesting

   The value of an established tag continues from the point the tag is
   embedded in text until either:

      A. The text itself goes out of scope, as defined by the
         application. (E.g. for line-oriented protocols, when reaching
         the end-of-line or end-of-string; for text streams, when
         reaching the end-of-stream; etc.)

   or

      B. The tag is explicitly cancelled by the CANCEL TAG character.

   Tags of the same type cannot be nested in any way. The appearance of
   a new embedded language tag, for example, after text which was
   already language tagged, simply changes the tagged value for
   subsequent text to that specified in the new tag.

Tags of different type can have interdigitating scope, but not
hierarchical scope. In effect, tags of different type completely
ignore each other, so that the use of language tags can be completely
asynchronous with the use of character set source tags (or any other
tag type) in the same text in the future.

4.5 Cancelling Tag Values

U-000E007F CANCEL TAG is provided to allow the specific cancelling of
a tag value. The use of CANCEL TAG has the following syntax. To
cancel a tag value of a particular type, prefix the CANCEL TAG
character with the tag identification character of the appropriate
type. For example, the complete string to cancel a language tag is:

U-000E0001 U-000E007F

The value of the relevant tag type returns to the default state for
that tag type, namely: no tag value specified, the same as untagged
text.

The use of CANCEL TAG without a prefixed tag identification character
cancels *any* Plane 14 tag values which may be defined. Since only
language tags are currently provided with an explicit tag
identification character, only language tags are currently affected.

The main function of CANCEL TAG is to make possible such operations
as blind concatenation of strings in a tagged context without the
propagation of inappropriate tag values across the string boundaries.
For example, a string tagged with a Japanese language tag can have
its tag value "sealed off" with a terminating CANCEL TAG before
another string of unknown language value is concatenated to it. This
would prevent the string of unknown language from being erroneously
marked as being Japanese simply because of a concatenation to a
Japanese string.

4.6 Tag Syntax Description

An extended BNF (Backus-Naur Form) description of the tags specified
in this proposal is found below. Note the following BNF extensions
used in this formalism:

1. Semantic constraints are specified by rules in the form of an
   assertion specified between double braces; the variable $$ denotes
   the string consisting of all terminal symbols matched by the this
   non-terminal.

   Example: {{ Assert ($$[0] == '?'); }}

              Meaning: The first character of the string matched by this
                         non-terminal must be '?'

   2. A number of predicate functions are employed in semantic
      constraint rules which are not otherwise defined; their name is
      sufficient for determining their predication.

        Example: IsRFC1766LanguageIdentifier (tag-argument)

        Meaning: tag-argument is a valid RFC1766 language identifier

   3. A lexical expander function, TAG, is employed to denote the tag
      form of an ASCII character; the argument to this function is
      either a character or a character set specified by a range or
      enumeration expression.

        Example: TAG('-')

        Meaning: TAG HYPHEN-MINUS

        Example: TAG([A-Z])

        Meaning: TAG LATIN CAPITAL LETTER A ...
                   TAG LATIN CAPITAL LETTER Z

   4. A macro is employed to denote terminal symbols that are character
      literals which can't be directly represented in ASCII. The
      argument to the macro is the UNICODE (ISO/IEC 10646) character
      name.

        Example: '${TAG CANCEL}'

        Meaning: character literal whose code value is U-000E007F

   5. Occurrence indicators used are '+' (one or more) and '*' (zero or
      more); optional occurrence is indicated by enclosure in '[' and
      ']'.

4.6.1 Formal Tag Syntax

tag   :   language-tag
                         | cancel-all-tag
                         ;

language-tag   :   language-tag-introducer language-tag-argument
                         ;

```
language-tag-argument   :   tag-argument
              {{ Assert ( IsRFC1766LanguageIdentifier ( $$ ); }}
                        |   tag-cancel
                        ;


cancel-all-tag          :   tag-cancel
                        ;


tag-argument            :   tag-character+
                        ;


tag-character           :   { c : c in
          TAG( { a : a in printable ASCII characters or SPACE } ) }
                        ;


language-tag-introducer :   '${TAG LANGUAGE}'
                        ;


tag-cancel              :   '${TAG CANCEL}'
                        ;
```


5.0 Tag Types

5.1 Language Tags

   Language tags are of general interest and should have a high degree
   of interoperability for protocol usage. To this end, a specific
   LANGUAGE TAG tag identification character is provided. A Plane 14
   tag string prefixed by U-000E0001 LANGUAGE TAG is specified to
   constitute a language tag. Furthermore, the tag values for the
   language tag are to be spelled out as specified in RFC 1766, making
   use only of registered tag values or of user-defined language tags
   starting with the characters "x-".

   For example, to embed a language tag for Japanese, the Plane 14
   characters would be used as follows. The Japanese tag from RFC 1766
   is "ja" (composed of ISO 639 language id) or, alternatively, "ja-JP"
   (composed of ISO 639 language id plus ISO 3166 country id). Since
   RFC 1766 specifies that language tags are not case significant, it is
   recommended that for language tags, the entire tag be lowercased
   before conversion to Plane 14 tag characters. (This would not be
   required for Unicode conformance, but should be followed as general
   practice by protocols making use of RFC 1766 language tags, to
   simplify and speed up the processing for operations which need to
   identify or ignore language tags embedded in text.) Lowercasing,

rather than uppercasing, is recommended because it follows the
majority practice of expressing language tag values in lowercase
letters.

Thus the entire language tag (in its longer form) would be converted
to Plane 14 tag characters as follows:

U-000E0001 U-000E006A U-000E0061 U-000E002D U-000E006A U-000E0070

The language tag (in its shorter, "ja" form) could be expressed as
follows:

U-000E0001 U-000E006A U-000E0061

The value of this string is then expressed in whichever encoding form
(UCS-4, UTF-16, UTF-8) is required and embedded in text at the
relevant point.

5.2 Additional Tags

Additional tag identification characters might be defined in the
future. An example would be a CHARACTER SET SOURCE TAG, or a GENERIC
TAG for private definition of tags.

In each case, when a specific tag identification character is
encoded, a corresponding reference standard for the values of the
tags associated with the identifier should be designated, so that
interoperating parties which make use of the tags will know how to
interpret the values the tags may take.

6.0 Display Issues

All characters in the tag character block are considered to have no
visible rendering in normal text. A process which interprets tags may
choose to modify the rendering of text based on the tag values (as
for example, changing font to preferred style for rendering Chinese
versus Japanese). The tag characters themselves have no display; they
may be considered similar to a U+200B ZERO WIDTH SPACE in that
regard. The tag characters also do not affect breaking, joining, or
any other format or layout properties, except insofar as the process
interpreting the tag chooses to impose such behavior based on the tag
value.

For debugging or other operations which must render the tags
themselves visible, it is advisable that the tag characters be
rendered using the corresponding ASCII character glyphs (perhaps
modified systematically to differentiate them from normal ASCII

characters). But, as noted below, the tag character values are chosen
so that even without display support, the tag characters will be
interpretable in most debuggers.

7.0 Unicode Conformance Issues

   The basic rules for Unicode conformance for the tag characters are
   exactly the same as for any other Unicode characters. A conformant
   process is not required to interpret the tag characters. If it does
   not interpret tag characters, it should leave their values
   undisturbed and do whatever it does with any other uninterpreted
   characters. If it does interpret them, it should interpret them
   according to the standard, i.e. as spelled-out tags.

   So for a non-TagAware Unicode application, any language tag
   characters (or any other kind of tag expressed with Plane 14 tag
   characters) encountered would be handled exactly as for uninterpreted
   Tibetan from the BMP, uninterpreted Linear B from Plane 1, or
   uninterpreted Egyptian hieroglyphics from private use space in Plane
   15.

   A TagAware but TagPhobic Unicode application can recognize the tag
   character range in Plane 14 and choose to deliberately strip them out
   completely to produce plain text with no tags.

   The presence of a correctly formed tag cannot be taken as a guarantee
   that the data so tagged is correctly tagged. For example, nothing
   prevents an application from erroneously labelling French data as
   Spanish, or from labelling JIS-derived data as Japanese, even if it
   contains Greek or Cyrillic characters.

7.1 Note on Encoding Language Tags

   The fact that this proposal for encoding tag characters in Unicode
   includes a mechanism for specifying language tag values does not mean
   that Unicode is departing from one of its basic encoding principles:

       Unicode encodes scripts, not languages.

   This is still true of the Unicode encoding (and ISO/IEC 10646), even
   in the presence of a mechanism for specifying language tags in plain
   text. There is nothing obligatory about the use of Plane 14 tags,
   whether for language tags or any other kind of tags.

   Language tagging in no way impacts current encoded characters or the
   encoding of future scripts.

It is fully anticipated that implementations of Unicode which already
make use of out-of-band mechanisms for language tagging or "heavy-
weight" in-band mechanisms such as HTML will continue to do exactly
what they are doing and will ignore Plane 14 tag characters
completely.

8.0 Security Considerations

There are no known security issues raised by this document.

References

    [ISO10646] ISO/IEC 10646-1:1993 International Organization for
               Standardization. "Information Technology -- Universal
               Multiple-Octet Coded Character Set (UCS) -- Part 1:
               Architecture and Basic Multilingual Plane", Geneva, 1993.

    [RFC1766] Alvestrand, H., "Tags for the Identification of
               Languages", RFC 1766, March 1995.

    [RFC2070] Yergeau, F., Nicol, G. Adams, G. and M. Duerst,
               "Internationalization of the Hypertext Markup Language",
               RFC 2070, January 1997.

    [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
               Requirement Levels", BCP 14, RFC 2119, March 1997.

    [RFC2130] Weider, C. Preston, C., Simonsen, K., Alvestrand, H.,
               Atkinson, R., Crispin, M. and P. Svanberg, "The Report of
               the IAB Character Set Workshop held 29 February - 1 March,
               1996", RFC 2130, April 1997.

    [UNICODE] The Unicode Standard, Version 2.0, The Unicode Consortium,
               Addison-Wesley, July 1996.

Acknowledgements

Authors' Addresses

    Ken Whistler
    Sybase, Inc.
    6475 Christie Ave.
    Emeryville, CA 94608-1050

    Phone: +1 510 922 3611
    EMail: kenw@sybase.com

    Glenn Adams
    Spyglass, Inc.
    One Cambridge Center
    Cambridge, MA 02142

    Phone: +1 617 679 4652
    EMail: glenn@spyglass.com

Full Copyright Statement