Doc Type	Working Group Document
Title	UCS Transformation Formats summary, non-error and error sequences
Source	Kent Karlsson
Status	Individual Contribution
Action	For consideration by JTC1/SC2/WG2
Date	2007-04-25

Introduction

This suggestion is related to SC2/WG2 N3248 (Synchronization Issues for UTF-8) and L2/07-116 (Unicode Security Exploit). This is a proposal on how to make UTF-8, UTF-16, and UTF-32 synchronised as well as specified for how to handle error sequences, which hitherto has not been fully specified. The proposal here allows for some flexibility, though, and maybe one could allow for more flexibility, keeping security issues in mind.

The formulations below are not exact text edits proposed, and I leave to the committees and editors to do more wordsmithing. Meta-remarks are given here in italics and indented.

Handling of error sequences for all of the UCS Transformation Formats

A sequence of one or more error sequences **shall** during conversion (including integrity checking) be replace by one or more SUBSTITUTE (U+001A) characters or abort the conversion (or integrity checking) with an error. A few code units that follow an error sequence can, but should not, be included in the error sequence.

> Note: An error sequence must **not** be replaced by nothing (the empty sequence of characters) nor by some other code sequence than for a non-empty sequence of SUBSTITUTE characters. It is implementation defined how long the sequence of SUBSTITUTE characters is for each error sequence.

MOTIVATIONS:
Replacing an error sequence by nothing or by something else than a sequence of SUB may cause a security problem.

The SUBSTITUTE C0 control code is specifically designated to mark character coding errors.

The number of SUBSTITUTE to use may vary depending on implementation. In some cases it may be convenient to replace an error sequence by e.g. the same number of SUBSTITUTEs as there are code units in the error sequence (e.g. if one is doing in-place consistency checking and replacement of error sequences), in other cases it may be more convenient to replace the entire error sequences by just one

SUBSTITUTE. Of course, the number of SUBSTITUTEs used for an error should not be excessive (so if needed, limit the number of SUBs to the number of code units in the error sequence).

If a few additional code units that follow an error sequence are included in the error sequence, that can lead to that proper following sequences are interpreted as error sequences as well, as the actual start of the proper sequence was included as in a preceeding error sequence).

The specification below uses regular expressions to give the normal and error sequences. E.g. [00-7F] matches any code with value between 00 and 7F (hexadecimal) inclusive, * denotes a sequence of zero or more, + denotes a sequence of one or more. They are given with a reference name on the left.

The reference names are not used formally, and can be deleted from the actual text that is to go into the respective standards texts.

UTF-8

Normal decodable sequences

8SEQ1 [00-7F]

8SEQ2 [C2-DF][80-BF]

8SEQ3A [E0][A0-BF][80-BF]
8SEQ3B [E1-EC][80-BF][80-BF]
8SEQ3C [ED][80-9F][80-BF]
8SEQ3D [EE-EF][80-BF][80-BF]

8SEQ4A [F0][90-BF][80-BF][80-BF]
8SEQ4B [F1-F3][80-BF][80-BF][80-BF]
8SEQ4C [F4][80-8F][80-BF][80-BF]

MOTIVATION
This is the already existing non-surrogates shortest form definition of UTF-8 from Unicode. But 10646 does not yet make this restriction to shortest form, so it still allows an old security problem with UTF-8.

Error sequences

8ERR1A [80-BF]+
8ERR1B [C0-FF][80-BF]*, but does not match any of the decodable sequences listed above

MOTIVATION
There is no reason to start counting number of continuation bytes/octets here. An implementation is free to use more than one SUB character for the error sequence, if conversion is not interrupted by the error, but one may set an upper limit on the nr of SUBs (see above).

Explicitly listing, as regular expressions, the error sequences of 8ERR1B is possible, but is intricate and error prone, so best avoided in the standards texts.

UTF-16

Normal decodable sequences

16SEQ1A [0000-D7FF]
16SEQ1B [E000-FFFF]

16SEQ2 [D800-DBFF][DC00-DFFF]

Error sequences

16ERR1 [DC00-DFFF]+
16ERR2 [D800-DBFF][DC00-DFFF]*, but does not match any of the decodable sequences listed above

MOTIVATION
Formulated as similarly as possible to the error situation for UTF-8.

UTF-32

Normal decodable sequences

32SEQ1A [00000000-0000D7FF]
32SEQ1B [0000E000-0010FFFF]

Error sequences

32ERR1 [0000DC00-0000DFFF]+
32ERR2 [0000D800-0000DBFF][0000DC00-0000DFFF]*, but does not match any of the decodable sequences listed above
32ERR3 [00110000-FFFFFFFF]

MOTIVATION
Formulated as similarly as possible to the error situation for UTF-8 and UTF-16. Otherwise one can simplify 32ERR1 and 32ERR2 to
32ERRALT1 [0000D800-0000DFFF]
which has no formal consequences given the formulations about error sequence handling given above.

For 10646, UTF-32 and UCS-4 have so far been regarded as equivalent. The proposal here is to replace UCS-4 by UTF-32 with the error cases specified above.