Developer Release Note - Text Encoding Converter Manager 1.3

(Draft 2, Dec. 1, 1997, P. Edberg)

This version of the Text Encoding Converter Manager (TEC) is included with Mac OS 8.1. The HFS Plus volume format introduced with Mac OS 8.1 stores filenames using the canonical decomposition form of Unicode 2.0; several of the TEC 1.3 changes are to support HFS Plus.

1. Interface file changes

a) Moved contents of old Unicode.h into new file UnicodeConverter.h to avoid confusion as other Unicode-related functionality is added over the next few months. Unicode.h includes UnicodeConverter.h, but is otherwise currently empty.

b) Added constant `kUnicodeCanonicalDecompVariant` (TextCommon.h) to specify a variant of Unicode using canonical decomposition (maximal decomposition with characters in canonical order). This constant has the same value as the constant `kUnicodeMaxDecomposedVariant` (unsupported in earlier versions of TEC). (The only other Unicode variant currently supported allows all defined Unicode characters, and is specified by the constant `kUnicodeNoSubset`).

c) Added constant `kUnicodeUseHFSPlusMapping` (UnicodeConverter.h), which can be used for the `mappingVersion` field of a `UnicodeMapping` structure to specify the mapping version used by HFS Plus. (The only other constant specifying a value for this field is `kUnicodeUseLatestMapping`).

d) Defined new feature/fix bits for the `tecUnicodeConverterFeatures` field of the `TECInfo` structure returned by `TECGetInfo`, to indicate new bug fixes/enhancements in TEC 1.3. These bits are:

`kTECTextRunBitClearFixBit` — `ConvertFromUnicodeToTextRun` & `ConvertFromUnicodeToScriptCodeRun` now function correctly if the `kUnicodeTextRunBit` is clear (previously their determination of best target encoding was incorrect).

`kTECTextToUnicodeScanFixBit` — `ConvertFromTextToUnicode` mappings can now depend on context and saved state. There are several related changes:

- Malformed input produces kTextMalformedInputErr.
- ConvertFromTextToUnicode accepts the control flags `kUnicodeLooseMappingsMask`, `kUnicodeKeepInfoMask`, `kUnicodeStringUnterminatedMask`.
- No redundant direction overrides when converting Mac OS Arabic & Hebrew to Unicode
- Improved mapping of 0x30-0x39 digits in Mac OS Arabic when loose mappings are used
- Better context-dependent mapping of certain characters in Mac OS Indic encodings.

e) Renamed the static library initialization and termination functions from `InitializeUnicode` and `TerminateUnicode` to `InitializeUnicodeConverter` and `TerminateUnicodeConverter`, for the same reasons as in (a) above. This should not be a problem, since we have not previously released a version of the static library. However, the old names are still exported by the static library just in case.

2. Implementation bug fixes

a) Fixed a crashing bug in `TECGetAvailableSniffers` which occurred in low-memory situations. (#1675195)

b) TECGetTextEncodingFromInternetName was sensitive to the case of the Internet name passed in, even though Internet names are supposed to be case-insensitive.

c) TECCreateOneToManyConverter and TECCreateOneToManyConverterFromPath should return paramError if the numOutputEncodings is 0.

d) Actually allowed the `kUnicodeStringUnterminatedBit` control flag to be used with `ConvertFromUnicodeToText` (`…ToTextRun`, `…ToScriptCodeRun`). This control was perviously documented for use with these APIs (and the supporting code was implemented). However, if it was used, the functions returned `paramErr`, due to an error in parameter validity checking. (#1684449)

e) Fixed problems with `ConvertFromUnicodeToTextRun` (`…ToScriptCodeRun`) when the `kUnicodeTextRun` control flag is clear; in this case these functions were making bad guesses about the best target encoding. (#1683393)

f) Fixed problems with handle locking and unlocking that showed up when multiple threads were calling the Unicode Converter. (#1684120)

g) `ConvertFromUnicodeToTextRun` (`…ToScriptCodeRun`) could enter infinite loop when running out of buffer space after executing a fallback handler. (#2005303)

h) Fixed several problems in `ConvertFromUnicodeToText` (`…ToTextRun`, `…ToScriptCodeRun`) for Unicode in UTF-8 format: Errors in direction resolution (and a crash when handling bidi text), and errors recovering from scanning ahead too far for text element boundaries.

i) In `ConvertFromUnicodeToText` (`…ToTextRun`, `…ToScriptCodeRun`), the `kUnicodeVerticalFormBit` was ignored when converting to any Japanese variant except `kJapaneseStandardVariant` and `kJapanesePostScriptScreenVariant`.

j) Fixed `CreateUnicodeToTextRunInfo` (and the `…ByEncoding` and `…ByScriptCode` forms) so that if 0 is passed for number of mappings/encodings/scripts or if NULL is passed for the array, it only creates entries for the script variants that are installed, instead of for all of the possible variants for each installed script.

3. Implementation enhancements

a) Allowed the following control flags to be used with `ConvertFromTextToUnicode` (previously, these were only intended for use with `ConvertFromUnicodeToText, …ToTextRun, …ToScriptCodeRun`): `kUnicodeLooseMappingsMask`, `kUnicodeKeepInfoMask`, `kUnicodeStringUnterminatedMask`.

b) Improved the `ConvertFromTextToUnicode` scanner to emit context-dependent information that can affect the mappings, and provided for saving scanner state in the TextToUnicodeInfo structure if `kUnicodeKeepInfoMask` is set. Enhanced the mapping table formats to permit mappings that depend on context information from the scanner and on tolerance information (i.e. whether loose mappings are requested).

c) When `ConvertFromTextToUnicode` encounters an invalid sequence of bytes in a particular encoding—such as 0x8120 in Shift-JIS—it now reports `kTextMalformedInputErr` (previously, it attempted to look up the invalid combination and returned `kTECUnmappableElementErr`).

d) The 68K static library version of the Unicode Converter & Text Common functionality is available with this release. In order to use it, the TEC 1.3 extension and the TEC 1.3 Text Encodings folder and files must be present.

4. Mapping changes

Among other things, the changes listed below ensure 100% roundtrip fidelity for strict mapping in either direction (non-Unicode to Unicoa and back or vice versa) for both Mac OS encodings and other non-Unicode encodings.

a) Added tables to support mapping between Mac OS encodings and the `kUnicodeCanonicalDecompVariant` variant of Unicode; the mapping version for these tables is `kUnicodeUseHFSPlusMapping`. These tables are located in the Text Encoding Converter extension itself, rather than in files in the Text Encodings folder (note that some of these tables also support other mappings, such as Shift-JIS, EUC-CN, Big-5, and EUC-KR). Note: This Unicode variant is currently not supported by the high-level Text Encoding Converter.

b) Eliminated redundant direction overrides when converting from Mac OS Arabic, Farsi or Hebrew to Unicode.

c) If ConvertFromTextToUnicode is called with the kUnicodeLooseMappings bit set, then the mapping of 0x30-0x39 digits in MacArabic and MacFarsi depends on the context, in a manner similar to how the WorldScript I display of these digits depends on context. If the 0x30-0x90 digits are preceded by Latin latters or other “strong European” characters, they display as “Western” digits in WorldScript I, and they are mapped to Unicode characters 0030-0039. Otherwise, they are displayed with the Arabic digits forms in WorldScript I, and they are mapped to the Unicode characters 0660-0669.

d) Fixed the scanners for EUC-CN and Big-5 to use the correct high-byte range.

e) In cases where we mapped to a Unicode character that has a one-character canonical decomposition, change the mapping to use the canonical decomposition. This affected the following mappings for Mac OS encodings:
   - Roman, Croatian, Icelandic, Turkish - 0xBD
   - Greek - 0xAF
   - Symbol - 0xE1 and 0xF1

f) Changed mapping of several characters in Mac OS Romanian (0xAF, 0xBF, 0xDE, 0xDF) to use COMBINING COMMA BELOW

g) Mapped the remainder of the user-defined range in MacJapanese and Shift-JIS

h) Defined several new corporate characters to be “grouping transcoding hints”: These precede a group of 2,3 or 4 standard Unicode characters that are to be treated as a group for transcoding. This way we can map additional characters that are in Apple character sets—but not in Unicode—using mostly standard Unicodes plus one of these transcoding hints. Using these, we changed the mapping for several characters in Mac OS encodings that previously just mapped to single corporate characters:
   - Japanese - 0x8591, 0x85AB-AD, 0x85BF-C1, 0x865D, 0x869E, 0x86CE, 0x86D3-D6, 0x87FB-FC
   - Hebrew - 0xC0
   - Farsi TrueType variant - 0xA4
   - Symbol - 0xE6-EE, 0xF4, 0xF6-FE

We also changed the mapping for several Mac OS Korean characters that were using two “combining disambiguation tag transcoding hints” to just use one. This affected 0xA14F-50, 0xA16A, 0xA170, 0xA198, 0xA19F, 0xA245-46, 0xA64E, 0xA78A, 0xA78E, 0xA792.

i) Other mapping fixes for Mac OS encodings:
- Devanagari - Deleted obsolete mappings for byte pairs 0xF0B5, 0xF0B8, 0xF0BF.
- Gurmukhi - Deleted mappings for byte pairs xB4E9, xB5E9, xBAE9, xBFE9, xC0E9, xC9E9. Changed the mapping for 0x91.
- Arabic AlBayan variant - 0x81 should be unmappable.
- Mac OS VT100 font encoding - Added mappings for 0xE2, 0xE3, 0xF5, 0xF6.
- Korean - Added mappings for many additional characters (these are all in the Apple extensions area)

j) Changed loose mapping of Unicode LINE SEPARATOR to be RETURN (instead of LINE FEED) for Mac OS encodings.

5. User interface changes

a) In Mac OS 8.1, the Text Encoding Converter extension is a required system component; the Extensions Manager will not allow it to be disabled, Finder will issue a warning if a user attempts to remove it, and a boot warning will be issued if it is not present.

b) Changed the Japanese name of Shift-JIS (returned by GetTextEncodingName) to have "Shift-JIS in romaji instead of katakana.