Unicode Transition:
Creating mapping files using Encore2Unicode.exe

Program Version 1.2
Lorna Priest, Bob Hallissy, et al.,
SIL Non-Roman Script Initiative (NRSI)

Encore2Unicode is a utility that helps you create a *draft* mapping description table from your Encore-derived font. After reading your font, Encore2Unicode generates a mapping description (either TECkit or UTR-22 XML format) and a report file (RTF).

The font must have been compiled using **TypeCaster** version 3 or greater. Encore2Unicode detects and will not process fonts from other sources or fonts built with an older version of TypeCaster. In the latter case, if the original CST file is available, you can re-build the font using the latest TypeCaster and then use Encore2Unicode.

It is important to note that Encore2Unicode can only provide a draft mapping description. It makes decisions based upon possible uses of glyphs in the Encore inventory that were considered most likely, but it is impossible to predict with complete confidence what Unicode characters might have been intended. It is essential, therefore, that the description generated by this utility be reviewed and revised before deployment

For a list of enhancements contained in version 1.2, see **Version History** at the end of this document.

Installation and use

The program is supplied as a self-extracting Zip file `encore2unicodeNN.exe` where *NN* is the Unicode version supported by this package. This installer defaults to unzipping all files into `C:\Program Files\SIL\Encore2Unicode`, but you may enter any folder name or browse for the folder of your choice.

***Special note for users of Microsoft Vista*:** In the default Vista configuration (i.e., UAC is on), in order for the self-extractor to create and put files into `C:\Program Files\SIL\Encore2Unicode`, it must be run with administrator privileges. Therefore, rather than double-clicking `encore2unicodeNN.exe` you should right-click it and select **Run as administrator**.

Encore2Unicode can be used by dragging files on to it using Windows Explorer, or via the command line interface. If you plan to use the command line interface, you may find it convenient to install the program into a folder that is named on your PATH. If you plan to use drag & drop, then you may find it helpful to create a shortcut to the Encore2Unicode.exe program on your desktop.

The files `Encore.txt`, `UnicodeData.txt`, `glyphlist.txt`, `blocks.txt`, `cp1252.txt` and `OverlayCompositions.txt` must be available either in your working directory or in the same directory as the Encore2Unicode program.

Uninstallation

Uninstallation is simple: just remove the folder into which the program was installed (default `C:\ Program Files\SIL\Encore2Unicode`).

How to use

Using Windows Explorer, you can drag your font onto `encore2unicode.exe` and a draft mapping description file (`<font>.map` or `<font>.xml`) and a report (`<font>.rtf`) will be created.

If you wish to use different options with Encore2Unicode, you can run it at the Command Prompt or create a shortcut to the .EXE and use the following syntax:

```
Encore2Unicode.exe [-d] [-n] [-q] [-x] ttfFile
```

Options:

- `-d` decompose *base+diacritic* where possible
- `-n` use Unicode character names instead of numbers
- `-x` output mappings in XML format (UTR-22)
- `-q` don't pause before exiting

Encore2Unicode automatically pauses (waiting for you to press the Return key) when it completes in order to give you time to read any messages from the screen. After you have read any messages, press Return key and the program will complete. If you do not want Encore2Unicode to pause, then you can use the –q option.

NB: The font need not be installed into Windows for Encore2Unicode to work. However, the font *does* need to be installed for the report file to display correctly.

Mapping Description File

You will need to make changes to the mapping description file that is generated. For quick reference, keep a copy of the Unicode 3.0 book handy. The report file that was generated will also prove to be a handy reference.

Encore2Unicode defaults to generating TECkit descriptions. The –x option enables UTR-22 XML format instead. For detailed information on the syntax of a TECkit mapping file, please read the documentation for TECkit. For detailed information about UTR-22 see http://www.unicode.org/unicode/reports/tr22/.

The remainder of these instructions assumes TECkit syntax.

Header

A TECkit description begins with a header containing various fields. Encore2Unicode will generate the following generic header template:

```
EncodingName            "(REG_ID)-<font-name>-(VERSION)"
DescriptiveName         ""
Version                 "0"
Contact                 "mailto:(YOUR_ADDRESS_HERE)"
RegistrationAuthority   "(REG_NAME)"
```

```
RegistrationName        "<font-name>-(VERSION)"

;*** Replace "(REG_ID)" with "SIL" or other organization identifier
;*** Replace "(REG_NAME)" with "SIL International" or other organization
name
;*** Replace "(VERSION)" with year the encoding was introduced
;*** Replace "(YOUR_ADDRESS_HERE)" with your email address
;*** Replace font name with other encoding identifier if appropriate

;set normalization flags only if you are sure they are appropriate
;SourceFlags     ()
;TargetFlags     ()
```

This header information needs to have informative information placed in between the quotes – you need to at least replace `(REG_ID)`, `(VERSION)`, `(YOUR_ADDRESS_HERE)` and `(REG_NAME)` with appropriate values.

Mappings

Basic review

Scan down the list of mappings to see if the mappings look accurate. Here is a typical mapping generated by Encore2Unicode:

```
0x79   <> U+0079 ; latin_small_letter_y  --  y = SILID 1033*
```

The comment field (everything after the semicolon) is made up of two parts separated by the double hyphen. The first part, in this case `latin_small_letter_y`, gives character identifier(s) based on the Unicode character name(s) of the character(s) from the right-hand side of the mapping (e.g., `U+0079`). The second part, e.g., `y = SILID 1033*`, shows the SILIDs of the components of the glyph found in the font.

An asterisk after an SILID indicates that there are additional comments related to the particular glyph used in the font. In the example above, the Encore glyph identified as “SILID 1033” was used, and a comment regarding that glyph is provided. You will find all such comments placed at the bottom of the mapping description file.

In the example above, the comment is:

```
;* SILID 1033: See also U+02B8 [Spacing Modifier Letters].
```

Encore2Unicode chose to map SILID 1033 to U+0079 (y), but another possible mapping is U+02B8 which is a “modifier letter small y” and can be found in the Spacing Modifier Letters block of Unicode. Encore2Unicode is not able to predict which Unicode character this glyph was actually used for in the Encore-generated font, so it proposes the option that was considered most probable and provides a comment to mention another possibility. In this particular example, you would probably want to keep the mapping which Encore2Unicode chose.

There may be other places where Encore2Unicode was not able to map the SILID code to a Unicode character. For example:

```
0x8D   <> PUA ; PUA  --  SILID 2915
```

SILID 2915 is a double barred “L” and this character does not exist in Unicode. In such instances you will need to request that the NRSI add the glyph to the PUA section of the Unicode Encore fonts and, if appropriate, propose it as an addition to the Unicode standard. Once the PUA codepoint is allocated, you will need to add that to the mapping file in place of “PUA”.

You may come across commented-out lines:

```
;0x7F  <>     ;   --  .notdef
```

This just means that codepoint wasn't defined in the font. No need to do anything.

Contextual rules

Another common situation that will need to be resolved is when multiple legacy characters map to the same Unicode character. For example, in the SIL IPA 93 fonts many of the diacritics have i-width/o-width and lowercase/uppercase variants. If Encore2Unicode is used on the SIL Doulos IPA93 font, the suggested mappings contain three that map to U+0304 COMBINING MACRON:

```
0x23 <> U+0304 ; combining_macron  --  SILID 0130 (space glyph ignored) + SILID 6009*
0x93 <> U+0304 ; combining_macron  --  SILID 0130 (space glyph ignored) + SILID 6109*
0x94 <> U+0304 ; combining_macron  --  SILID 0130 (space glyph ignored) + SILID 6109*
```

These rules will work fine for converting legacy data to Unicode, but they are problematic for converting Unicode data back to legacy: which mapping should be used? The correct choice depends on the immediately preceding character, i.e. on the context. The TECkit mapping description language supports contextual mappings; the current UTR-22 standard does not (although extensions to the standard have been proposed that would provide this functionality). For example, one might rewrite the above rules as:

```
0x23 <> combining_macron / [lc_o_width] _
0x93 <> combining_macron / [uc_o_width] _
0x94 <> combining_macron / [uc_i_width] _
```

This assumes that you also add class definitions for uppercase and lowercase i- and o-width characters. See the TECkit documentation for further details.

Component positioning

If you are familiar with SIL IPA, you are probably wondering where the lowercase o-width macron is. Looking at the SIL IPA documentation indicates that it should be at character 220 (0xDC), but Encore2Unicode suggested the following mapping for 220:

```
0xDC   <>  PUA ; PUA  --  SILID 0130 (space glyph ignored) + SILID 6418*
```

It looks as though for the lowercase i-width macron the CST author used SILID 6418 (4-unit bar through lower case) and then employed glyph positioning within the CST to move the glyph up to the right place for a macron.

This leads to an important limitation of Encore2Unicode: it doesn't look at the position of glyphs within a character. Rather, it looks only at the SILID of the glyphs used. In this case the suggested mapping is clearly wrong, but we know that only because we know something about the legacy encoding. The correct mapping would probably be something like:

```
0xDC  <> combining_macron / [lc_i_width] _
```

Compiling

After making all the necessary changes you should now be able to compile this mapping description file with **TECkit** and then use the resulting mapping file to convert your text files to Unicode.

Report File

Encore2Unicode generates a report file that provides a visual representation of the mapping. In order to view the report file, *make sure that the legacy font you are working with is already installed on your system.* The report color-codes the mappings to help you identify areas that need attention. The scheme is as follows:

- Blue indicates Encore2Unicode believes this is not a standard Windows character (i.e. it does not correspond to codepage 1252). You should carefully review all of these mappings to make sure they do what you want. (Every mapping needs to be checked, but these are the ones where errors in the proposed draft are most likely to occur.)
- Red indicates there is no Unicode equivalent for the character. You should discuss these characters with your entity's Unicode expert or the NRSI to make decisions regarding PUA assignments and/or Unicode proposals.
- Grey indicates the character was not used in the font.
- Black indicates Encore2Unicode believes this is a standard Windows character, i.e., one from codepage 1252.

Currently there is a problem with viewing the report file in Word 2002; see **Known Problems and Weaknesses** below.

Potential Error Messages:

Mappings cannot be generated for <fontfilename>.

This font was built with an old version of the Encore font compiler. If the CST is available, you may be able to regenerate the font using the current version of TypeCaster, and then generate mappings from that version.

This does not appear to be an Encore-derived font.

Encore2Unicode will not work on this font.

Unable to load font <fontfilename> at Encore2Unicode.pl line 59.

Make sure you supplied the font's correct filename.

WARNING: font has been modified with FontMonger.

Although the font is a valid Encore-generated font, it has been modified using the font editing utility FontMonger. Depending on the changes made to the font, Encore2Unicode may or may not have trouble.

unrecognized glyph name '<glyphname>' for byte code <legacy codepoint>

Encore2Unicode depends on the internal glyph names assigned to the various characters by TypeCaster. This message indicates that the font has been modified and this name is not recognizable.

can't read <tablename> table!

Most likely the file you supplied to Encore2Unicode isn't a TrueType file.

Known Problems and Weaknesses

An important part of the Encore font system is that it permits the user to move glyphs up, down or sideways from their original default position. This feature is used, for example, to accurately place diacritics over their base characters. Encore2Unicode does not look at the position of any glyphs, and so can make some pretty severe blunders. For example a horizontal bar glyph such as SILID 9022 (hyphen dash) could be used to build a hyphen or minus sign, a baseline rule, a macron under, or a macron over by adjusting its position. Encore2Unicode does not see these position changes and its suggested mappings assume the default glyph positions.

Word (from Microsoft Office) has trouble with the report files; specifically the formatting for the second column (“Glyph”) may have random character attributes applied such as italic, bold, outline or strikethrough. One workaround is to open the file in WordPad instead of Word. If you want to use Word, after opening the report select Format Styles, select the character style named “Legacy”, and adjust the font formatting as needed.

Encore2Unicode does not attempt any validation of the mapping as a whole. For example, it does not detect situations where multiple legacy characters map to the same Unicode value. In such cases you may want to consider adding contextual rules so that the reverse mapping process (from Unicode data to legacy data) works correctly.

Version History

The Encore2Unicode *package* consists of the `Encore2Unicode.exe` program plus various data files that drive the program. With each new version of Unicode, newly standardized characters may be better choices for characters found in Encore fonts. Therefore, when appropriate, updated versions of the package have been released to correspond with updates to the Unicode standard. Since 2003 the version number of the Encore2Unicode package has indicated the version of Unicode that the package implements.

The `Encore2Unicode.exe` *program* is currently at version 1.2:

Version 1.2

Certain sequences of combining marks weren’t handled properly.

Now reads overlay composition data from an external file.

This version was initially packaged with data files for Unicode 5.1.

Version 1.1

Unless the new –q option is specified, Encore2Unicode always pauses and waits for the Enter key. This makes it easier to see warning messages when using drag&drop execution.

Mapping file header text is revised and there are additional helpful comments in the header area.

Use of Unicode's “combining overlay” marks (U+0334 - U+0338, U+20E5) is “discouraged” by suggesting PUA. (However, there are special exceptions for those few Unicode characters such as “barred-u” that someone might have composited using TypeCaster.)

A few mappings were fixed, and the comments were cleaned up – mostly in recognition that Unicode 3.2 is now released rather than pending.

The 0xB7 <> U+00B7 mapping is now colored black rather than blue in the report.

A number of glyphs will have corrected postscript names (in the comments and in the report).

Wherever the “U+” notation is used in the comments, the block-name is now included in square braces for reference.

The default glyph for 0xAD now maps correctly to U+00AD.

The .RTF output shows the correct pathname for the file.

Version 1.0

Original implementation.