Unicode in PNG — Why and How

Roland Fumey
SIL PNG

4 May 2007

This document covers the following Topics:

- *Why Unicode?*
- *In what cases do you need to convert your data to Unicode?*
- *What are the tools available, and how do you install and use them?*
- *Troubleshooting*

Note: *This document is meant to be updated. So you might want to check for a newer version sometime. Suggestions for additions and clarifications are welcome.*

Table of Contents

1 Why Unicode?

As you are aware, a number of special characters are being used in PNG that were not part of “standard” (ANSI) fonts on computer systems. Because of this, some of our colleagues did a good job replacing unneeded characters in ANSI fonts with those characters needed in PNG, thus creating the “PNG SIL fonts”, like *PNG SILDoulos*, *PNG SILSophia Lit*, etc. For example, they replaced “¢” with “ē”, “§” with “Ʒ”, “š” with “Ʉ”, etc. as it wasn’t possible at that time to actually add new characters to fonts. (The system was limited to 256 characters – of which many are undisplayable control characters.)

Whereas this worked quite nicely most of the time, the system was not without hiccups: For example, when using a PNG font and typing three dots “...” in Word, suddenly the character “ǐ” would spring up. Or, when writing phonetic data, the line would break in the middle of a word instead of moving the entire word to the next line. In addition, some people using non-English versions of Windows had troubles getting these characters displayed correctly at all. This all is because those fonts were ‘**hacked**’. The font system was not meant to be used like that!

Things got worse when the Euro sign (€) was introduced: To accommodate for this new character, Microsoft hijacked the codepoint x80, which is “Λ” (turned-V) in the PNG fonts, so that now many programs display the Euro sign there, **even** if you **have** the **correct** fonts installed! (There’s a similar story with “Ɨ” (I double-bar, x8E) which is often displayed as “Ž”, and “Ḡ” (G bar, x9E) which is often displayed as “ž”.)

> **Important**: Even with the correct fonts, you might not be able to read your old data[1] any more in 10 years’ time, as newer versions of applications and operating systems might not be able to handle hacked fonts!

The solution to this is, of course, **Unicode**, as Unicode assigns to each character its **unique codepoint**.

In the past, a codepoint like *xBC* was very ambiguous:

- In ANSI (codepage 1252) it was “¼” (one-quarter)[2]
- in the PNG fonts it was “ˇ” (caron)
- in SILIPA it was “ ̰ ” (tilde below)
- in some Greek fonts – but not in others! – it was “ ΐ ” (iota with diaeresis and acute)
- etc.

but in Unicode each of these characters has its own unique codepoint and name:

- “¼” is U+00BC VULGAR FRACTION ONE QUARTER
- “ˇ” is U+030C COMBINING CARON
- “ ̰” is U+0330 COMBINING TILDE BELOW
- “ ΐ ” is composed of U+03B9 GREEK SMALL LETTER IOTA, U+0308 COMBINING DIAERESIS and U+0301 COMBINING ACUTE ACCENT[3]

> **In Summary:**
>
> Unicode provides a unique number for every character,
> no matter what the platform,
> no matter what the program,
> no matter what the language.

(Quoted from http://www.unicode.org/standard/WhatIsUnicode.html)

Therefore:

- You should convert your old data to Unicode (see next chapter in what cases this is necessary)
- All your new files and documents should be **Unicode encoded**, **especially if you start a new project!**

[1] ‘**Old** data’ refers to **non-Unicode data**, which is often also called **Legacy** data or **Byte-encoded** data.

[2] This paper uses a number of different fonts and encodings, which hopefully should be embedded in the document and therefore displayed properly. If they don’t …, well, then this is just another illustration why we do need Unicode!

[3] Many accented characters are available in both composed and decomposed forms. See discussion in section 5.1, page 16.

Note: When starting a new project, **try to stick to the already available Unicode characters** if at all possible, as it easily takes several years to get a new character into Unicode and the committee wants really good evidence for its need before they approve it!

1.1 UTF-8, UTF-16, UTF-32

Although in Unicode each character has its unique codepoint, there are different ways how actual Unicode data is stored: UTF8, UTF16BE, UTF16LE, UTF32BE, and UTF32LE (where UTF stands for *Unicode transformation format,* BE for *Big Endian* and LE for *Little Endian*).

Most applications (including all SIL programs) seem to use **UTF8**, which is also the format that uses the least disk space. (Some programs actually use internally another UTF format although they read and write UTF8 data, but you don't have to worry about that …).

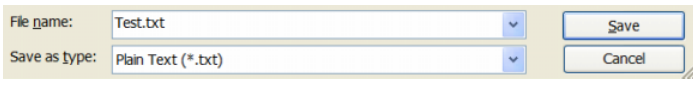
Therefore: Whenever an application gives you a choice of how to save a file (e.g. Windows XP's Notepad) **choose UTF8**.
(Do <u>not</u> choose 'Unicode', as what they mean by 'Unicode' is actually UTF16.)

One notable exception to this is if you are intending to use your file in **InDesign**. Then you must save your file as **UTF-16**.

1.1.1 And a note about <u>Word</u> in particular:

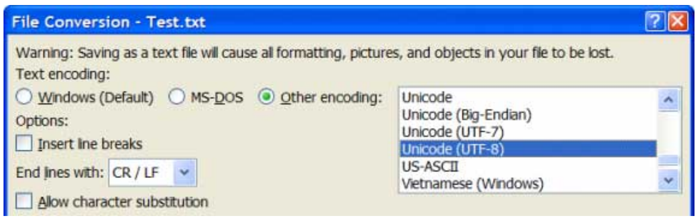
If you convert a text file in Word and need to save it as **text** file again, make sure that you save it in the correct format, as follows:

- When the *Save As* dialog comes up, choose **Plain Text** in *Save as Type*:



File name: Test.txt
Save as type: Plain Text (*.txt)

- When the *File Conversion* dialog comes up, choose **<u>O</u>ther encoding** and then select **Unicode (UTF-8)** (not just 'Unicode', as this would save the file as UTF16LE)



File Conversion - Test.txt

Warning: Saving as a text file will cause all formatting, pictures, and objects in your file to be lost.

Text encoding:
( ) Windows (Default) ( ) MS-DOS [✓ Other encoding:]

Unicode
Unicode (Big-Endian)
Unicode (UTF-7)
[✓ Unicode (UTF-8)]
US-ASCII
Vietnamese (Windows)

Options:
☐ Insert line breaks
End lines with: CR / LF
☐ Allow character substitution

Note: This only applies to saving a file as (unformatted) text file in Word. When saving a Word document as a 'normal' *.doc* file (or *.rtf* or *.xml*) you don't need to (and you can't) tell Word in what Unicode transformation format the file should be saved.

2 The Good News!

2.1 When do you not need to convert your data …

If you don't have any special characters in the language you're working with, i.e. if all your characters are in the range **"a-z/A-Z"** you don't need to do any conversion. All your language files are already in Unicode! The code points of these characters remain the same.

In addition, **in MS Word documents** (as of Word 97) **'normal' European characters** (like À, à, á, Á, â, Â, ã, Ã, ä, Ä, å, Å, Æ, æ, ç, Ç, È, è, É, é, Ê, ê, Ë, ë, ì, Ì, Í, í, î, Î, Ï, ï, Ð, ð, ñ, Ñ, ò, Ò, ó, Ó, Ô, ô, õ, Õ, ö, Ö, Ø, ø, ß, Ù, ù, Ú, ú, û, Û, ü, Ü, etc) **are automatically converted to Unicode by Word itself**. So you don't have to convert such Word documents, unless they also contain other special characters in 'hacked' fonts like the PNG fonts.

Other programs might also automatically convert your input of 'normal' European characters, but you will have to make sure that the file is **saved in Unicode** (in Windows XP Notepad, for example, by choosing **Encoding: UTF-8** when doing *Save As*).

2.1.1 However

If a file with 'normal' European characters **can't be saved as Unicode**, it needs to be converted.

If you use any **other characters** in **'hacked' fonts like the PNG fonts** you definitely need to convert your data to Unicode.

You should also convert your data if you use **colons** or **apostrophes** in words (e.g. for lengthening or glottal stops). See more about that in section 5.2, page 16.

Any papers you wrote that include non-Unicode **IPA characters** (e.g. OPDs and Phonology papers) should be converted to Unicode too!

2.2 Unicode 5.0

Second good news: As of **Unicode 5.0** (and with it the Doulos/Charis SIL Fonts, version 4.100) **almost all special characters that we use in PNG** are now available in **Standard Unicode**!

The **only two characters** not yet in standard Unicode and thus remaining in the **Private Use Area (PUA)** of SIL Unicode fonts for now are:

ꞌ	MODIFIER LETTER STRAIGHT APOSTROPHE, also called: LATIN LETTER DOTLESS EXCLAMATION or SALTILLO	U+F21D
꞉	MODIFIER LETTER COLON	U+F1E9

Both of these characters have been accepted into Unicode, but no definite codepoints have been assigned yet (so they are not yet part of Unicode 5.0).

Since these two characters are probably only used by very few language teams in PNG, this means that most teams can now safely convert their data to Unicode (except maybe for literacy, but see about that in section 3.1.1.1, page 8).

2.2.1 Converting from older versions of Unicode to Unicode 5.0

The advent of Unicode 5.0 (unfortunately) means that you might have to convert your data again if you used an earlier version of the Doulos SIL and Doulos Charis fonts, as some of our PNG characters had not previously been accepted into standard Unicode and therefore were only available in SIL's Private Use Area (PUA) of Unicode.

Specifically, you will need to convert your Unicode data to Unicode 5.0, if you use one or more of the following PNG characters:

Ʉ	LATIN CAPITAL LETTER U BAR	U+0244	(before: U+F218)[4]
Ʌ	LATIN CAPITAL LETTER TURNED V	U+0245	(before: U+F219)
Ƚ	LATIN CAPITAL LETTER L WITH BAR	U+023D	(before: U+F21F)
Ɫ	LATIN CAPITAL LETTER L WITH MIDDLE TILDE	U+2C62	(before: U+F242)
Ⱡ	LATIN CAPITAL LETTER L WITH DOUBLE BAR	U+2C60	(before: U+F20F)
ⱡ	LATIN SMALL LETTER L WITH DOUBLE BAR	U+2C61	(before: U+F20E)
Ɋ	LATIN CAPITAL LETTER SMALL Q WITH HOOK TAIL	U+024A	(before: U+F212)

[4] All characters with codepoints starting with F1, F2 and F3 are PUA characters.

ʠ	LATIN SMALL LETTER Q WITH HOOK TAIL	U+024B	(before: U+F211)
ɂ	LATIN SMALL LETTER GLOTTAL STOP	U+0242	(before: U+F21E)

And in case you happened to use a pre-Unicode 4 SIL font, you would also need to convert your data again if you used the following character:

a͞a	COMBINING DOUBLE MACRON (shown with vowels)	U+035E	(before: U+F177)

You might also have to convert your older Unicode data to Unicode 5.0 in files using **IPA characters**.

For converting from an earlier version of Unicode to Unicode 5.0, use **SILPUAtoUnicode50.tec** (in *C:\Program Files\Common Files\SIL\MapsTables*) or choose **SIL PUA 3.2<>UNICODE 5.0** in Word's Data Conversion macro (installation and usage described further down in this document).

> **Note:** The fact that you might have to convert from older versions of Unicode to Unicode 5.0 does **not** mean that Unicode itself is changing! It only means that **additions** have been made to it. In other words, characters that SIL previously needed to put into the PUA (Private Use Area) of Unicode fonts have now become standard Unicode, and therefore we should now use those standard codepoints. This will also allow you to use many other Unicode fonts, not just SIL fonts (as long as those other fonts contain all the characters you need.)

3 Available Tools

Note: All SIL tools listed here are available on the **SIL PNG LCORE FTP site** ftp://ftp.sil.org.pg/Departments/LCORE/Software (also accessible through the **LCORE website**) or internationally on NRSI's website http://scripts.sil.org (unless stated otherwise).

3.1 Unicode Fonts

To be able to display Unicode data correctly, **Unicode compatible fonts** are needed.

Note: In general, SIL fonts whose names **start** with "**SIL**" or "**PNG**" (e.g. *SIL Galatia*, *SILDoulosIPA*, *PNG SILCharis*, etc.) are **legacy** (i.e. non-Unicode) fonts.

Fonts whose names are **followed** by "**SIL**" are **Unicode** fonts (e.g. *Doulos SIL*, *Galatia SIL*, *Ezra SIL*, etc.), with the exception of *Gentium*, *Andika* and *SIL Yi*, which are Unicode fonts too.

3.1.1 SIL Roman Unicode fonts

For **Roman** script, including **all characters and diacritic combinations we need here in PNG**, you can use the **Doulos SIL** or **Charis SIL** fonts.

Important: Make sure you have **version 4.100 or later**, as earlier versions of these fonts are not Unicode 5.0 compatible!

Note: These fonts also contain **all IPA characters**. Thus we do not need any special fonts any more for IPA!

(There used to be a font called *SIL Unicode IPA*. This one should no longer be used, as it is **not** Unicode 5.0 compatible and has been superseded by and incorporated into Doulos SIL!)

The **Charis SIL** font system has separate font files for **regular**, **bold**, **italic** and **bold italic** (*CharisSILR.ttf*, *CharisSILB.ttf*, *CharisSILI.ttf* and *CharisSILBI.ttf*),

Doulos SIL, however, consists of only a **regular** font (*DoulosSILR.ttf*). You can still display the Doulos font in bold, italic and bold italic, but it might not look as nice.

In general the *Charis* fonts might look nicer in print, and the *Doulos* fonts nicer on screen.

3.1.1.1 Unicode Literacy fonts

- **Literacy versions** of the **Doulos** and **Charis** Unicode fonts (with the right kind of "ɑ" and "ɡ") are available on the SIL PNG LCORE FTP site. The fonts have been created using the *TypeTuner* program (see section 3.4.3, page 11).

  Note: Earlier literacy versions of the Charis and Doulos fonts (in *CharisSILLit4.0.02.r1.zip* and *DoulosSILLit4.0.14.r1.zip*) are not Unicode 5.0 compatible and should no longer be used.

In addition, work on Unicode **literacy fonts** is currently in progress as follows:

- NRSI[5] is working on the **Andika** literacy font, which should hopefully be released sometime in 2007.

  Note: The Andika fonts we received up to January 2007 are only **review** versions. They don't contain all the characters we need in PNG and are Unicode 4.1, not 5.0

- Unicode versions of our "**good old**" **PNG fonts** (*PNG SILSophia Lit*, *PNG SILSophia CQLit*, etc.) might become available, depending on the state of the SIL *Reprise* program.

3.1.2 Non-SIL Roman Unicode fonts

If you find **other** (non-SIL) **Unicode fonts** that have all the characters you need for your PNG language (and display all characters-diacritics combinations correctly), feel free to use them!

If they are truly Unicode compatible this should work without a problem.

[5] *NRSI* (which stands for *Non Roman Script Initiative*) is SIL International's research and development team that deals with all sorts of font and encoding issues. Fortunately – despite their name – they don't do this only for people who have Non-Roman scripts needs, but for us too!

3.1.3 Greek and Hebrew Unicode fonts

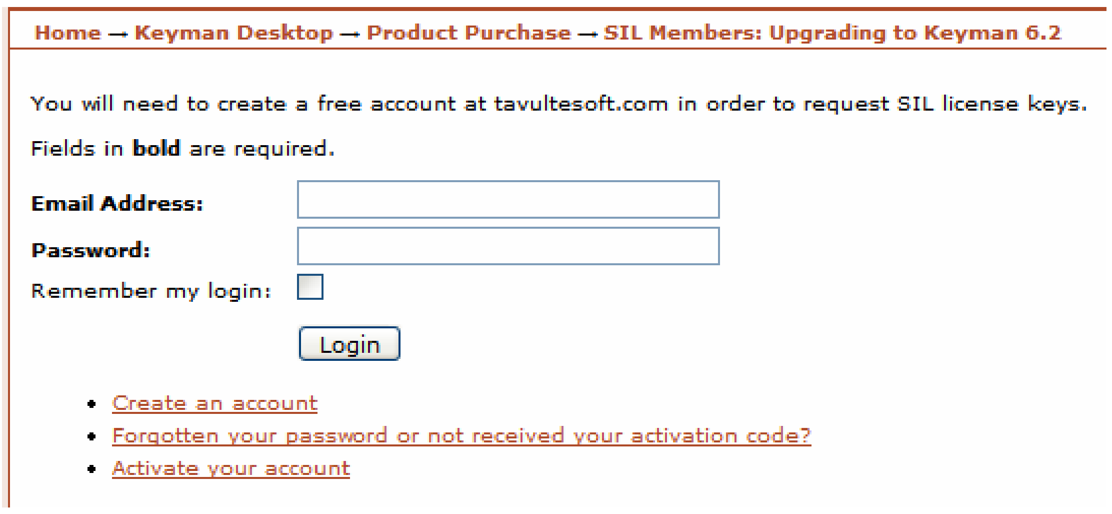
Any files with **Greek or Hebrew** data should use Unicode Greek and Hebrew fonts. The SIL Greek Unicode font is called **Galatia SIL** (version 2.0.1 or later) and the SIL Hebrew Unicode font is called **Ezra SIL** (version 2.0 or later). These fonts also come with Keyman definition files (in the zip files) that enable you to type these fonts.

3.2 Unicode Input Tools

To be able to type special characters in Unicode, you will need Keyman 6.2 or later with a Unicode compatible **Keyman** definition file or a keyboard definition file written with the free **Microsoft Keyboard Layout Creator** program.

3.2.1 Keyman 6.2 or later

Keyman 6.2 is currently still free for SIL teams, but you will need to **register with Tavultesoft**.
On http://www.tavultesoft.com/buy/, under *Other Purchase Options*, there is a special entry **SIL Members - version 6.2**. When you click on it you will come to a web page like this:



Home → Keyman Desktop → Product Purchase → SIL Members: Upgrading to Keyman 6.2

You will need to create a free account at tavultesoft.com in order to request SIL license keys.

Fields in **bold** are required.

Email Address:

Password:

Remember my login:

- Create an account
- Forgotten your password or not received your activation code?
- Activate your account

They will also want to know how many copies you will use.

Note: For Windows VISTA you will need Keyman version 7 (or later), which is no longer free to SIL members. NRSI is currently in contact with Tavultesoft about a corporate license.

3.2.2 Microsoft Keyboard Layout Creator (MSKLC)

As an alternative that should work for many kinds of simple key definitions, you can also use the free **Microsoft Keyboard Layout Creator** (MSKLC), Version 1.3.4073 or later.

For Windows Vista you will probably need Version 1.4 or later. This version should also work with Windows XP.

For version 1.3 you will need to install version 1.0 or 1.1 of the **.NET Framework** first. Version 1.4 requires .NET Framework v2.0.

Both versions of the Keyboard Layout Creator and of the .NET Framework (also called MS DotNet Framework) are available on the LCORE FTP site, or from http://www.microsoft.com/globaldev/tools/msklc.mspx and http://msdn2.microsoft.com/en-us/netframework/aa731542.aspx, respectively.

Note: Earlier versions of **Toolbox** had problems accepting input from keyboards built with MSKLC. This problem has been **solved** in **Toolbox 1.5**.

About **Paratext** NRSI writes: “Paratext will not, and probably never will, accept input from keyboards built with MSKLC.” My personal experience is that MSKLC in Paratext works fine for some (but not all) simple characters, but that there are definite problems with the placement of diacritics. So it might or might not work for you.

3.2.3 Keyboard definition files

3.2.3.1 For your PNG language

For your specific PNG language it's probably most efficient (in terms of your typing) to write a specific Keyman or Microsoft Keyboard Layout definition file.

3.2.3.2 For IPA

NRSI wrote a **Keyman** file for typing IPA. Make sure you have the newest Unicode 5.0 compatible version (1.1.1 or later). The definition file and documentation is available on the LCORE intranet or from http://scripts.sil.org/UniIPAKeyboard.

A **Microsoft Keyboard Layout** definition file for IPA is also available from the same locations.

3.2.3.3 For Greek and Hebrew

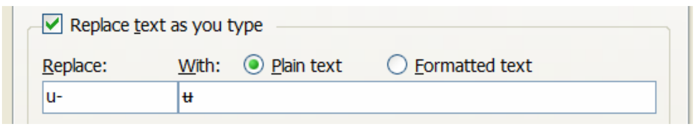
The SIL Greek and Hebrew Unicode fonts (*Galatia SIL* and *Ezra SIL*) come with Keyman definition files (in their respective zip files). Use those for typing these languages.

3.2.4 Input options in Word

If you need to enter a specific Unicode character only sporadically, you can enter it in Word 2003 (and probably some other versions too) by typing the code, selecting it, and then pressing **Alt+X**.

For example, to insert "ʉ" (u-bar), type its Unicode code point *0289*, select it and then hold down the ALT key and press X.

You can also make such a character permanent in Word by selecting it, then choosing **Tools – AutoCorrect Options**. It will then appear in the *With* column under *Replace text as you type*. In the *Replace* column you can then enter whatever character combination you would like to use to type this character.


☑ Replace text as you type

Replace: u-
With: [✓ Plain text] Formatted text
ʉ

Note, however, that a character defined like that will only work in Word. You should use Keyman or MSKLC if you plan to use special characters in other programs too.

3.3 Conversion Tools

3.3.1 SILConverters

SILConverters (version 2.5 or later) contains just about everything you need for the conversion of data to and from Unicode, except that the SIL PNG conversion mapping is not yet included in version 2.5.

Apart from the NRSI and LCORE PNG websites, SILConverters is also available on the **SIL Language Freeware BTE disc 1** (LST_07-011 or later), that comes with **Word & Deed**.

> **Note:** All programs and files that were previously part of a separate **TECkit** package are now **included** in SILConverters!

3.3.2 SIL PNG conversion mapping

The SIL PNG conversion mapping (**version 2.01 or later)**, consisting of the **SIL-PNG.map** (source-code) and **SIL-PNG.tec** (compiled) is available on the LCORE intranet and from http://scripts.sil.org/ConversionMaps (in *SILPNGUnicodeConversion_v2.01.zip*).

This mapping (as most others) not only allows you to convert from Legacy to Unicode, but also the other way round, if this is ever needed.

> **Note:** Future versions of SILConverters should include the PNG mapping files, so you won't have to download and install them separately!

> **Note:** The SIL PNG mapping file is written for the conversion of **PNG SIL fonts** (like *PNG SILDoulos*, *PNG SILDoulos*, etc.) to Unicode. If you use another kind of encoding, either adjustments need to be made to the SIL-PNG.map or an entirely new mapping must be written. (One option to do the latter is UBS's c2u program described below.)

> **Important:** If you use any special characters that are **not** in the PNG SIL fonts and/or can't find them in the SIL Unicode fonts, **we need to hear from you**, so that we can make sure they get into Unicode. Note, however, that the Unicode committee will want really good evidence for its need before they add a new character! Try to stick to the already available Unicode characters, if possible.

3.3.3 UBS's c2u (Convert to Unicode Tool)

UBS's **c2u** has been specifically written for the conversion of Paratext files, but it can also be used to convert other files. It allows you to write your own mapping and, as an additional benefit, will also create a Keyman keyboard definition for your needs.

This program is especially useful if you need to convert from any encoding system for which no mapping file exists yet.

> **Note:** For the conversion from PNG SIL encoded data to Unicode you do **not** need this program, as a conversion mapping is already available (unless you really want to reinvent the wheel …)

3.4 Other Tools

3.4.1 Unicode Word Macros

NRSI wrote a few Macros (*UnicodeWordMacros151.zip*) that can be useful for dealing with Unicode in Word documents. These include providing a means to display the Unicode value of any character, to enter any Unicode character and to search for any Unicode character.

The Macros can be accessed in Word with the following Toolbar buttons:

3.4.2 PNG Unicode Macros

The Macro in the *PNGUnicodeMacros* template allows you to check a Word document after converting it to Unicode to see if really everything has been converted or if there is any legacy font left, either in the document itself or in the styles definition.

After installing it – see the accompanying Readme file how to do this – you should find a new entry under *Tools*, called *PNG Check For Legacy Fonts*. Just click on it to run the Macro.

The Macro does not make any changes to your documents. It just tells you what it finds.

3.4.3 TypeTuner

This (currently being tested) SIL program will allow us to create customized fonts by choosing which of the glyph variants should be the default in any SIL Unicode font.

Thus, we will, for example, be able to create literacy versions of the Doulos SIL and Charis SIL fonts with the right kinds of "ɑ" and "g" (already done!) and other versions of these fonts that contain your preferred glyph for Eng and Glottal stops, for example.

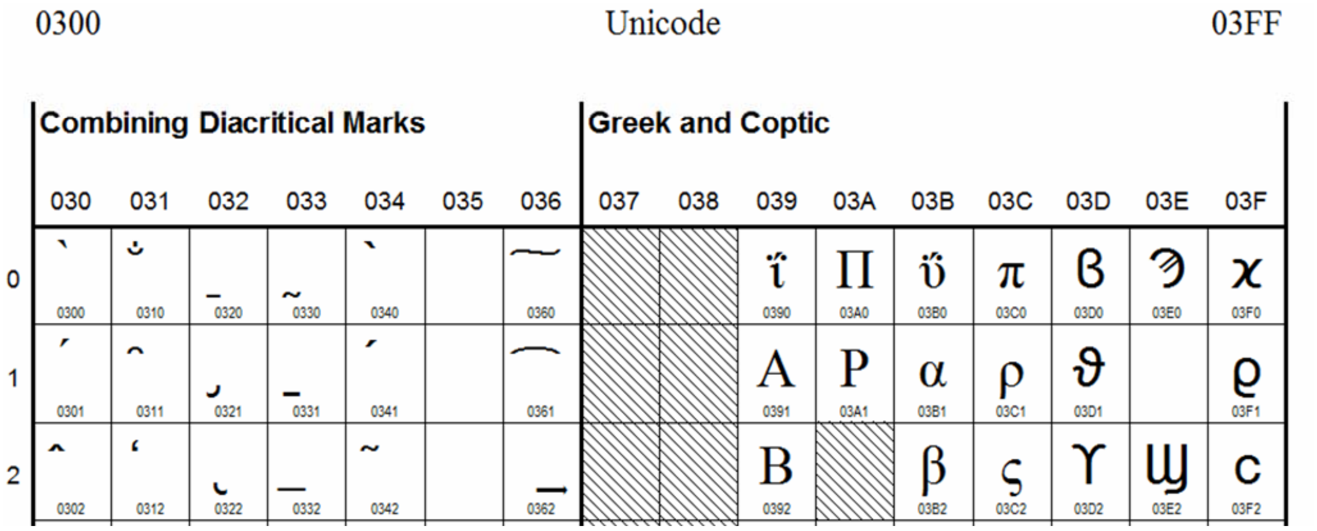
> **Note:** For SIL programs that are based on SIL's Graphite rendering system, e.g. FieldWorks and WorldPad (in contrast to Word, Notepad, Paratext, etc. which are based on Microsoft's Uniscribe rendering system), such customized fonts are not needed, as you can choose your preferred glyph right in the program.

3.4.4 Unibook™ Character Browser

The Unibook™ Character Browser is a tool designed to present information about the characters defined in the Unicode Standard. With it you can find codepoints, character names and other useful information about many Unicode characters.

It is available from the LCORE FTP site or from http://www.unicode.org/unibook/.

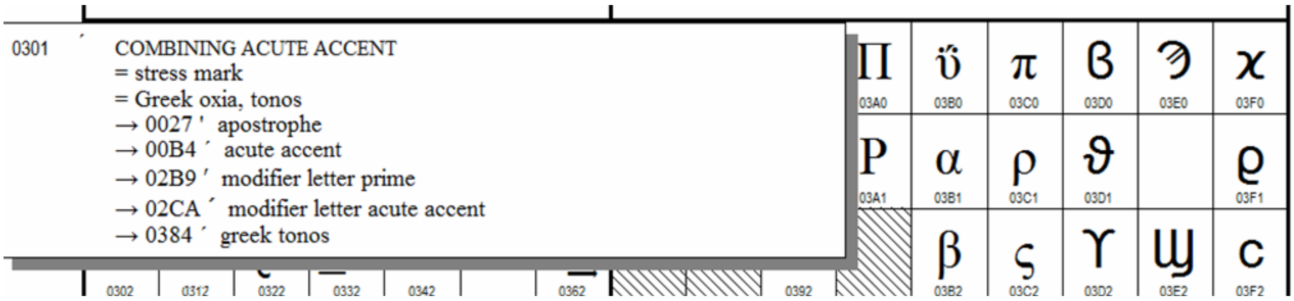
Unibook sample:



0300 Unicode 03FF

	Combining Diacritical Marks							Greek and Coptic								
	030	031	032	033	034	035	036	037	038	039	03A	03B	03C	03D	03E	03F
0	◌̀ 0300	◌̐ 0310	◌̠ 0320	◌̰ 0330	◌̀ 0340		◌͠ 0360	▨	▨	ΐ 0390	Π 03A0	ΰ 03B0	π 03C0	ϐ 03D0	ϡ 03E0	ϰ 03F0
1	◌́ 0301	◌̑ 0311	◌̡ 0321	◌̱ 0331	◌́ 0341		◌͡ 0361	▨	▨	Α 0391	Ρ 03A1	α 03B1	ρ 03C1	ϑ 03D1		ϱ 03F1
2	◌̂ 0302	◌̒ 0312	◌̢ 0322	◌̲ 0332	◌͂ 0342		◌͢ 0362	▨	▨	Β 0392	▨	β 03B2	ς 03C2	ϒ 03D2	Ϣ 03E2	ϲ 03F2

Clicking on a character gives you more information about it, e.g.

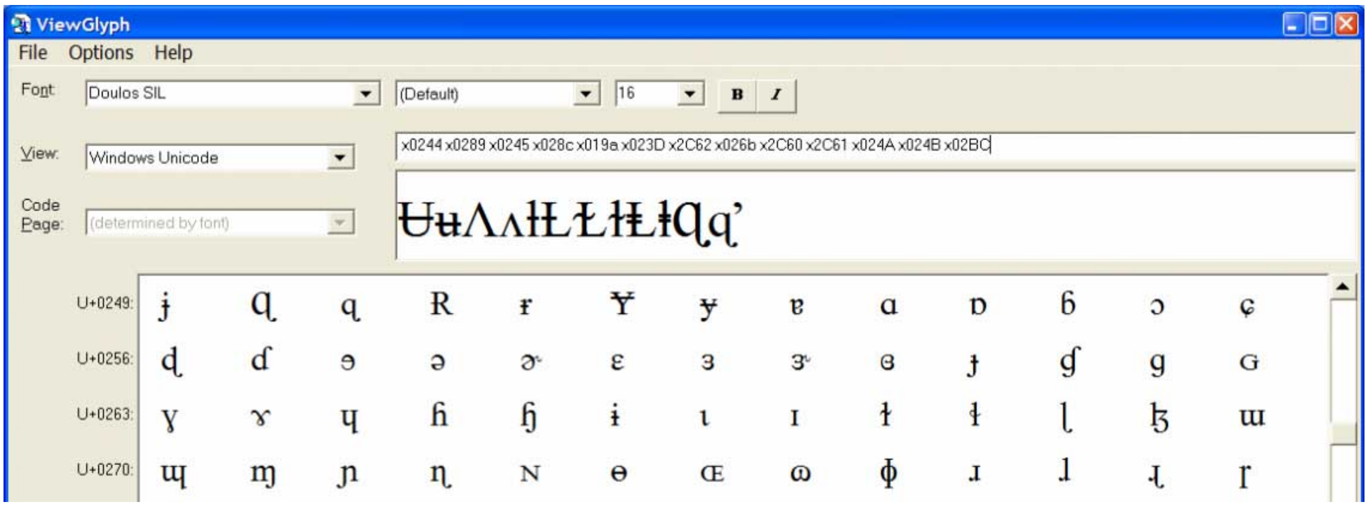


0301 ́ COMBINING ACUTE ACCENT
= stress mark
= Greek oxia, tonos
→ 0027 ' apostrophe
→ 00B4 ´ acute accent
→ 02B9 ʹ modifier letter prime
→ 02CA ˊ modifier letter acute accent
→ 0384 ΄ greek tonos

3.4.5 *ViewGlyph*

SIL's ViewGlyph program (version 1.77.0 or later) allows you to view the glyphs plus a lot of other information about any font.

ViewGlyph sample:



Font: Doulos SIL (Default) 16

View: Windows Unicode

Code Page: (determined by font)

x0244 x0289 x0245 x028c x019a x023D x2C62 x026b x2C60 x2C61 x024A x024B x02BC

ɄʉɅʌƚȽⱢɫⱠⱡɊɋʼ

U+0249:	ɉ	Ɋ	ɋ	Ɍ	ɍ	Ɏ	ɏ	ɐ	ɑ	ɒ	ɓ	ɔ	ɕ
U+0256:	ɖ	ɗ	ɘ	ə	ɚ	ɛ	ɜ	ɝ	ɞ	ɟ	ɠ	ɡ	ɢ
U+0263:	ɣ	ɤ	ɥ	ɦ	ɧ	ɨ	ɩ	ɪ	ɫ	ɬ	ɭ	ɮ	ɯ
U+0270:	ɰ	ɱ	ɲ	ɳ	ɴ	ɵ	ɶ	ɷ	ɸ	ɹ	ɺ	ɻ	ɼ

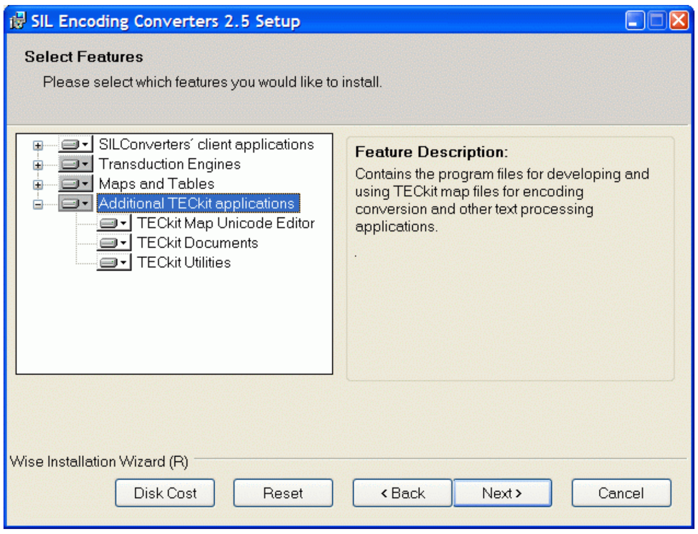
4 Installation and Setup

4.1 *Installing Fonts*

- If the fonts are zipped, extract them to a temporary location first.
- Then
  - (in Windows XP) open the Fonts control panel (Start, Control Panel, Appearance and Themes, then look in the upper left – or if you use classic Control Panel view: look for the 'Fonts' folder).
  - or open the fonts folder directly: C:\WINDOWS\FONTS
- Then drag the font files (e.g. *CharisSILR.ttf*, *DoulosSILR.ttf*, etc.) into the Font window – or choose File, Install New Font..., and navigate to your font files.
- **Note:** For installing a new version of a font you need to delete the old version in the Font folder first.

4.2 *Installing SILConverters*

- Install SILConverters directly from the latest SIL Language Freeware BTE disc (LST_07-011 or later) or run *SetupEC.msi*
- Choose destination folder *C:\Program Files\SIL\SILConverters* as recommended by the installation program
- When the **Select Features** dialog comes up, leave the ones that are chosen by default and **add** (under *Additional TECkit applications*):
  - **TECkit Map Unicode Editor**
  - **TECkit Documents**
  - **TECkit Utilities**


SIL Encoding Converters 2.5 Setup

Select Features
Please select which features you would like to install.

- SILConverters' client applications
- Transduction Engines
- Maps and Tables
- Additional TECkit applications
  - TECkit Map Unicode Editor
  - TECkit Documents
  - TECkit Utilities

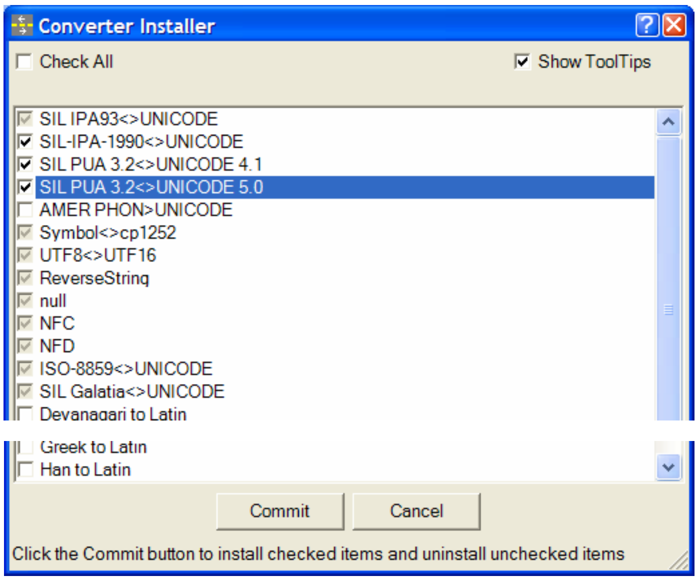
Feature Description:
Contains the program files for developing and using TECkit map files for encoding conversion and other text processing applications.

Wise Installation Wizard (R)

- Later in the installation process, when '**Converter Installer**' comes up, leave the already checked entries as they are, and add:
  - SIL-IPA-1990<>UNICODE
  - SIL PUA 3.2<>UNICODE 4.1
  - SIL PUA 3.2<>UNICODE 5.0

  (and whatever else you might be interested in)

  Note: Later versions of SILConverters should also include **SIL-PNG<>Unicode**. When that is the case, add this one too (of course)!


Converter Installer

☐ Check All ☑ Show ToolTips

☑ SIL IPA93<>UNICODE
☑ SIL-IPA-1990<>UNICODE
☑ SIL PUA 3.2<>UNICODE 4.1
☑ SIL PUA 3.2<>UNICODE 5.0
☐ AMER PHON>UNICODE
☑ Symbol<>cp1252
☑ UTF8<>UTF16
☑ ReverseString
☑ null
☑ NFC
☑ NFD
☑ ISO-8859<>UNICODE
☑ SIL Galatia<>UNICODE
☐ Devanagari to Latin
☐ Greek to Latin
☐ Han to Latin

Click the Commit button to install checked items and uninstall unchecked items

- Click on '**Commit'**

  (Converters can also be added or removed later without going through the whole installation process.)

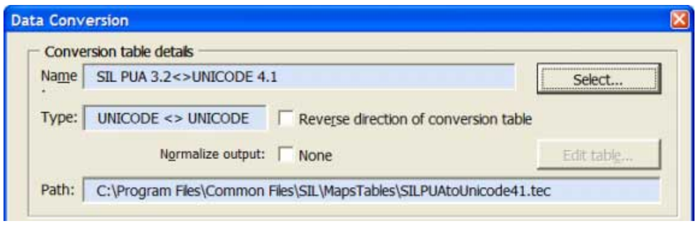
Note: All programs installed by SILConverters can be run from the **Start Menu, All Programs: SIL Converters.**

4.3 Adding the SIL-PNG mapping to SILConverters

Unless the SIL-PNG mapping files are already included in SILConverters, you need to add them as follows:

Copy the files **SIL-PNG.map** and **SIL-PNG.tec** from their zip file (*SILPNGUnicodeConversion_v2.01.zip*) to wherever you feel appropriate. (It might be best, however, to copy them into the folder *C:\Program Files\Common Files\SIL\MapsTables* where all other mapping files are. But if you do that you need to make sure that you make backup copies if you ever modify them.)

- In Word (after having installed *SILConverters*), click on **Tools** – **Data Conversion**, which opens the 'Data Conversion' dialog (with probably other details than on this screenshot)
- There, click on **Select**



Data Conversion

Conversion table details

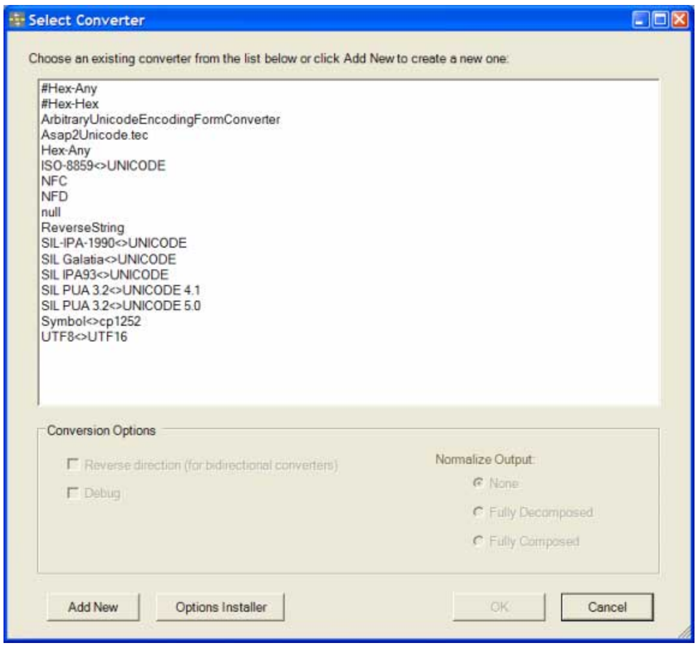
Name: SIL PUA 3.2<>UNICODE 4.1 — Select...

Type: UNICODE <> UNICODE — ☐ Reverse direction of conversion table

Normalize output: ☐ None — Edit table...

Path: C:\Program Files\Common Files\SIL\MapsTables\SILPUAtoUnicode41.tec

- This opens the **Select Converter** dialog.
- If **SIL-PNG<>Unicode** is not already on the Converters list, click on **Add New**



Select Converter

Choose an existing converter from the list below or click Add New to create a new one:

```
#Hex-Any
#Hex-Hex
ArbitraryUnicodeEncodingFormConverter
Asap2Unicode.tec
Hex-Any
ISO-8859<>UNICODE
NFC
NFD
null
ReverseString
SIL-IPA-1990<>UNICODE
SIL Galatia<>UNICODE
SIL IPA93<>UNICODE
SIL PUA 3.2<>UNICODE 4.1
SIL PUA 3.2<>UNICODE 5.0
Symbol<>cp1252
UTF8<>UTF16
```

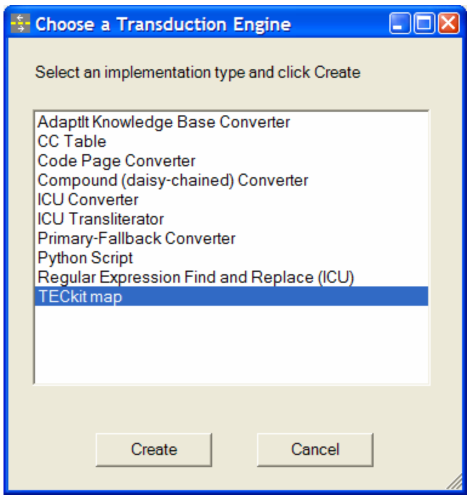
Conversion Options

☐ Reverse direction (for bidirectional converters)

☐ Debug

Normalize Output: [● None] ○ Fully Decomposed ○ Fully Composed

- In **Choose a Transduction Engine dialog** click on **TECkit map**,



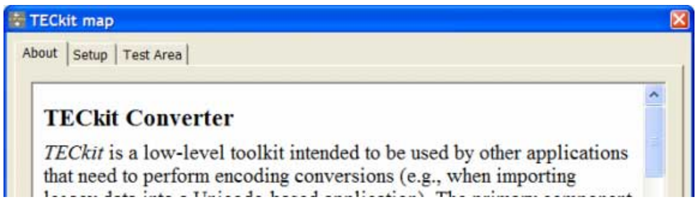
Choose a Transduction Engine

Select an implementation type and click Create

```
AdaptIt Knowledge Base Converter
CC Table
Code Page Converter
Compound (daisy-chained) Converter
ICU Converter
ICU Transliterator
Primary-Fallback Converter
Python Script
Regular Expression Find and Replace (ICU)
TECkit map
```

- Then click on **Create**

- Information about TECkit map comes up (which you can read if you are interested)

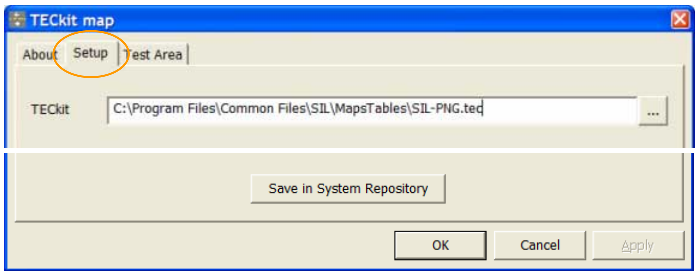


TECkit map

TECkit Converter

TECkit is a low-level toolkit intended to be used by other applications that need to perform encoding conversions (e.g., when importing

- Click on the **Setup** tab
- Now enter the path of your **SIL-PNG.tec** file. (or click on the "**…**" button to go to where you have the file)
- Click on **Save in System Repository**, to save SIL-PNG.tec as a permanent converter.

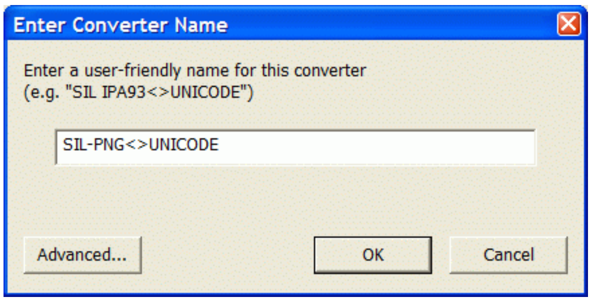


TECkit: C:\Program Files\Common Files\SIL\MapsTables\SIL-PNG.tec

Save in System Repository

- Enter a **user-friendly name** for this converter, e.g. '**SIL-PNG<>UNICODE**'

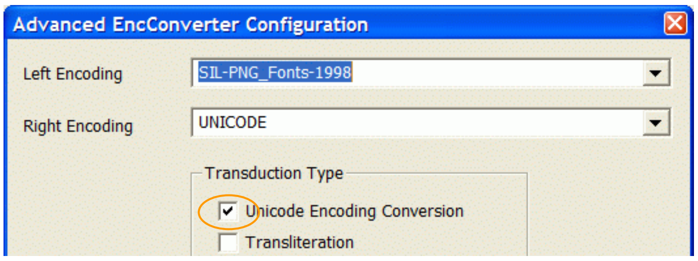
- Click on **Advanced**



Enter a user-friendly name for this converter
(e.g. "SIL IPA93<>UNICODE")

SIL-PNG<>UNICODE

- In **Advanced EncConverter Configuration** leave the two suggested entries unchanged and **check Unicode Encoding Conversion**
- Click on **OK** until you are back to the *Select Converter* dialog.



Left Encoding: SIL-PNG_Fonts-1998

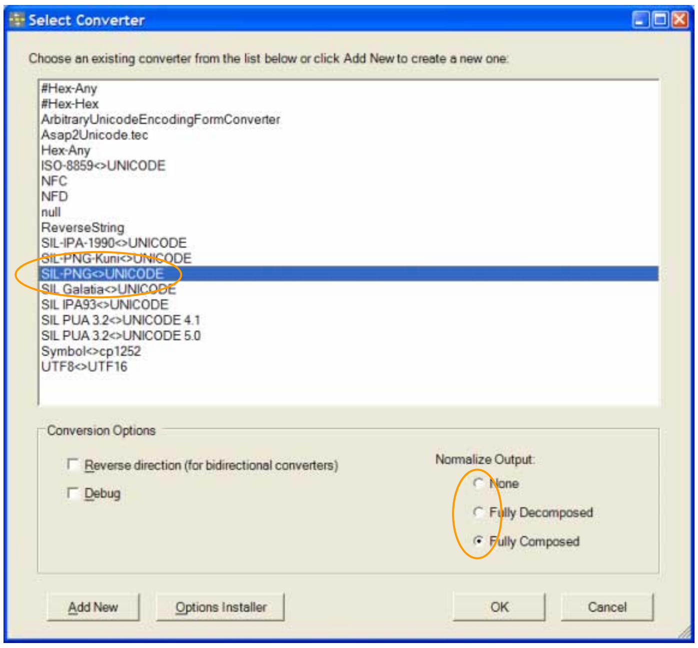
Right Encoding: UNICODE

Transduction Type

[✓ Unicode Encoding Conversion]

Transliteration

- 'SIL-PNG<>UNICODE' has now been added to the list of available converters.



Choose an existing converter from the list below or click Add New to create a new one:

#Hex-Any
#Hex-Hex
ArbitraryUnicodeEncodingFormConverter
Asap2Unicode.tec
Hex-Any
ISO-8859<>UNICODE
NFC
NFD
null
ReverseString
SIL-IPA-1990<>UNICODE
SIL-PNG-Kuni<>UNICODE
SIL-PNG<>UNICODE
SIL Galatia<>UNICODE
SIL IPA93<>UNICODE
SIL PUA 3.2<>UNICODE 4.1
SIL PUA 3.2<>UNICODE 5.0
Symbol<>cp1252
UTF8<>UTF16

Conversion Options

Reverse direction (for bidirectional converters)

Debug

Normalize Output: None / Fully Decomposed / [✓ Fully Composed]

- Choose your preferred Normalization under **Normalize Output** (See discussion about that in section 5.1, page 16.) This can easily be changed later if needed.
- Click **OK**
- Back in the *Data Conversion* dialog click **Cancel** to return to Word.

4.4 Installing UBS's c2u (Convert To Unicode Tool.exe)

To install this program, double click on the **Convert_to_Unicode_program_1.0.11.pud** patch file (For this to work you need to have Paratext 6 installed before.)

This then opens a command window as follows:

```
Portions (c) Bspatch 2003 Colin Percival.

Press Y<Enter> to apply patch X:\...\Convert_to_Unicode_program_1.0.11.pud.
Press N<Enter> to exit without applying patch.
```

When you press Y and Enter, the patch file installs the program **Convert To Unicode Tool.exe** and other necessary files in the *Paratext 6* folder under *Program Files* (normally *C:\Program Files\Paratext 6*). When this is done, make a **shortcut** to this program on your desktop or wherever most useful.

5 Things to consider

5.1 *If you have accented characters – about Normalization*

When converting from Legacy to Unicode you have **three Normalization options**: **None**, **Fully Decomposed** (also called: **NFD**), **Fully Composed** (also called: **NFC**). This tells the converter whether it should build characters with diacritics as one single character or as base characters plus separate diacritics.

For example, “é” is a composed character (codepoint 00E9), whereas “é” is decomposed (codepoints 0065 and 0301). Both versions should look exactly the same! (If they don’t, there is a problem either with the font or the rendering system. See Troubleshooting, section 7.1, page 25).

If you choose:

- **None:** Whenever you have a character in composed/decomposed form in the old encoding, this will convert it to (pretty much) the same in Unicode.
- **Fully Decomposed (NFD):** All characters will be decomposed, e.g. “ǘ” consists of three parts (codepoints 0075, 0308, and 0301)
- **Fully Composed (NFC):** All characters will be fully composed, **if they exist as such in Unicode**, e.g. “ǘ” is a single composed character (codepoint 01D8), but since “ö́” does not exist as a single character in Unicode, it will stay decomposed.

Recommendation:

- Use **Fully Decomposed** if you want to be able to **add/delete diacritics individually** from the base characters. This is especially useful for **grammatical tone**.
- If you don’t want or need this, then you might want to choose **Fully Composed**.
- **None** is **not recommended** because you might have inconsistent data in that respect in your legacy data (i.e. the same character sometimes in composed and other times in decomposed form), which would then stay that way.

> **Note:** Whatever form you choose, make sure that your keyboard definition (created with **Keyman** or **MS Keyboard Layout**) inputs the characters in the same way!

If you ever change your mind about Normalization, you can always easily convert your files between NFC and NFD later (see section 6.1.7, page 19).

5.2 *If you use colons or apostrophes/quotes as word forming characters…*

If you use **colons** (e.g. for lengthening or vowel quality) or **apostrophes** (e.g. as glottal stops) **within words**, you should convert your data to Unicode, even if you don’t have any other special characters.

This is because Unicode makes a distinction between word forming characters and punctuation marks. Colons (:) and apostrophes (' " ‘ “”) are strictly to be used as punctuation marks, and not as orthographic characters.

- The word forming counterpart to the colon is the **MODIFIER LETTER COLON** (꞉). Unfortunately, as of Unicode 5.0, this character is not yet part of standard Unicode. It has, however, been accepted by the Unicode committee and should hopefully become standard Unicode soon! The character is currently available in the PUA (Private Use Area) of SIL Fonts (codepoint F1E9).
- For glottal stops the following characters (among others) are available:
  - The **MODIFIER LETTER APOSTROPHE** (ʼ) (codepoint 02BC). This letter also exists in a larger form, but there are currently only very few applications (e.g. *FieldWorks* and SIL’s *WorldPad*) that allow you to select glyph variants. We should, however, soon be able to build fonts with the larger glyph as default, once the TypeTuner program (see section 3.4.3, page 11) is available.
  - The **MODIFIER LETTER STRAIGHT APOSTROPHE** (ꞌ). This character is currently only available in the PUA of SIL Fonts, but it has been accepted and should hopefully become standard Unicode soon.

> **Note:** If the MODIFIER LETTER COLON and/or the MODIFIER LETTER STRAIGHT APOSTROPHE are the only special characters you have, you might want to wait with converting your data until they become available in standard Unicode.

Depending on your choice you might need to make a couple of changes in the SIL-PNG.map file. (See further comments in the *SIL-PNG.map* file.)

5.3 Capital Eng

In Unicode the two shapes of Capital Eng “Ŋ” (xDD) and “Ŋ” (xDE) are **variant glyphs of the same character** (codepoint 014A). In the SIL fonts, “Ŋ” is the default. The other glyph can be chosen by applications that allow glyph selection or customized fonts can be built with TypeTuner (see section 3.4.3, page 11).

When converting back from Unicode to Legacy (if you ever need to do that), you might have to adapt the SIL-PNG.map file if it converts to the ‘wrong’ Eng. (See further comments in the mapping file.)

6 Conversion to Unicode

For the Conversion from legacy data to Unicode we need to distinguish between files with a **single encoding** and such with **multiple encodings**:

Single-encoded means that a file has only **one encoding**. An example of a single encoded file is one that uses only the PNG SIL encoding (which uses any of the *PNG SIL …* fonts), and does not also have data in another encoding, e.g. SIL IPA or an old Greek font, etc.

Multi-encoded means that a file contains data in several kinds of legacy encodings, e.g. your vernacular data (in PNG SIL encoding), plus IPA and/or Greek and/or Hebrew.

> **Note:** A strictly **Unicode encoded** file is **always single-encoded**, even if it has all kinds of different scripts (like PNG special characters, European characters, Greek, Hebrew, Chinese, Thai, etc.)

6.1 Available converters and mapping files

The following lists and describes some of the more relevant converters and mapping files.

In Word's **Data Conversion macro** you select the appropriate converter through the *Select* button, then choosing the one you need (e.g. *SIL-PNG<>UNICODE*).

In **other converter programs** you refer to the appropriate converter through its *.tec* mapping file (normally residing in *C:\Program Files\Common Files\SIL\MapsTables*), e.g. *SIL-PNG.tec*.

6.1.1 Converting PNG fonts

To convert files with PNG characters based on the *PNG SIL fonts*, use **SIL-PNG<>UNICODE** (mapping file: **SIL-PNG.tec**).

> **Note:** Since the legacy PNG SIL fonts are based on the Windows Codepage 1252, they also contain many (but not all) European special characters.
>
> This means that a file containing vernacular data (in SIL PNG encoding) and any of European characters "Á, á, À, à, Â, â, Ä, ä, Ã, ã, Æ, æ, É, é, È, è, Ê, ê, Ë, ë, Í, í, ì, Î, î, Ï, ï, Ñ, ñ, Ó, ó, Ò, ò, Ô, ô, Ö, ö, Õ, õ, Ø, ø, Œ, œ, Ú, ú, Ù, ù, Û, û, Ü, ü" or any of the following signs "–, —, ', ', ", ", «, », ¤, ¬, °, ¶, •" can be treated as *a single-encoded* file.
>
> In addition, the SIL-PNG mapping file (as of version 2.01) also allows you to convert the characters "Å, å" and signs "‹, ©, ·" together with your vernacular data, as these are unused in the PNG SIL fonts.
>
> Besides that, you can adjust the SIL-PNG mapping file to also convert the German sharp-s "ß" if you don't have any curly glottal "ˀ" in your language or the sign "›" if you don't use the u-bar "ʉ". (See relevant entries and comments in SIL-PNG.map)

6.1.2 Converting SIL-IPA

The **SIL IPA93<>UNICODE** and **SIL-IPA-1990<>UNICODE** (mapping files: **silipa93.tec** and **SIL-IPA-1990.tec**[6]) can be used to convert files with IPA characters. Most of the files in PNG probably used the IPA93 "standard".

Speech analyzer files, however, used a different encoding, called ASAP (mapping file: Asap2Unicode.tec). The mapping will be included in the upcoming Speech Analyzer, version 3.

6.1.3 Converting older versions of Unicode

To convert files with characters that used to be only available in the PUA (Private Use Area) and have now become standard Unicode (see section 2.2, page 6, if this applies to you), use **SIL PUA 3.2<>UNICODE 5.0** (mapping file: **SILPUAtoUnicode50.tec**).

6.1.4 Converting 'normal' European characters

To convert files with standard European characters (also called Windows Codepage 1252), use **ISO8859<>UNICODE** (mapping file: **iso-8859.tec**)

6.1.5 Converting Greek

To convert files with Greek characters based on the old *SIL Galatia* encoding, use **SIL Galatia<>UNICODE** (mapping file: **SILGreek2004-04-27.tec**).

[6] SILConverters installs all mapping files in *C:\Program Files\Common Files\SIL\MapsTables*

6.1.6 Converting Hebrew

To convert files with Hebrew characters based on the old *SIL Ezra* encoding, use the mapping file **SILHebrewSE.tec** that comes in the zip file of the Ezra SIL 2 fonts. (See also further instructions there.)

6.1.7 Converting between NFD and NFC

To convert between Fully Decomposed and Fully Composed form (see discussion in section 5.1, page 16), select **NFD** or **NFC**, respectively, in the *Data Conversion macro*.

For text files[7], you can do this with *txtconv.exe* (see section 6.2.2 below). Just run it with no mapping file, specifying a Unicode encoding form for both input and output, and give it the normalization option you want.

6.1.8 Other conversion needs

If there are no converters/mapping files available for your needs you can

- either **write your own**, either with **UBS's c2u conversion tool**, i.e. *Convert To Unicode Tool.exe* (which might be the easiest way) or with the **TECkit Map Unicode Editor**
- or **adapt one of the existing mapping files** with the **TECkit Map Unicode Editor**. Just double click on a *.map* file to open it with this editor. After making the necessary changes you need to **compile** the file in the editor by pressing *Ctrl+K* (or choosing: *File – Compile*). This will create the *.tec* file needed for the conversion.

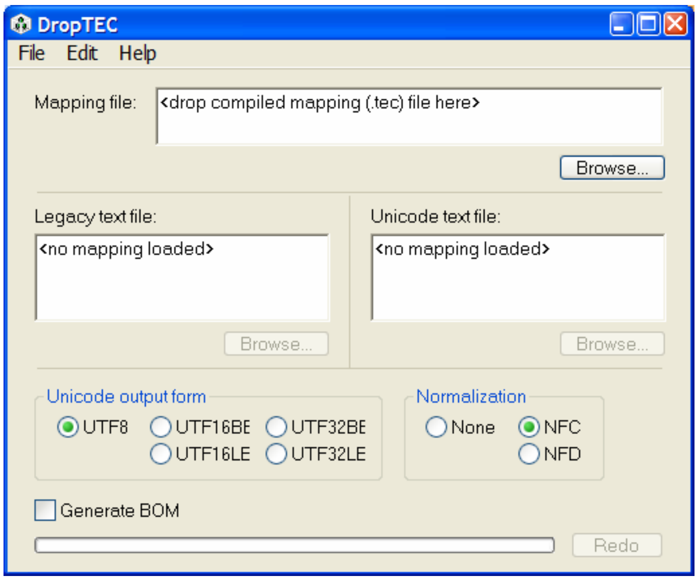
6.2 Converting Single-encoded text files

Text files[7] with a single encoding, e.g. Paratext files, can easily be converted in two different ways:

6.2.1 Using DropTEC for one or a few single-encoded text files

If you installed the *TECkit Utilities* as part of *SILConverters*, you should find **DropTEC** in the **Start Menu: SILConverters – TECkit – DropTEC**. (If you use it regularly, you might also want to make additional shortcuts to it wherever you prefer.)

- **Start DropTEC**
- Drag the **SIL-PNG.tec**[8] file into the *Mapping file* box
- Choose **UTF8** as Unicode output form
- Choose the **Normalization** type you want (see discussion in section 5.1, page 16)
- About **Generate BOM**: In most cases it doesn't matter if it is generated or not, but there might be some older programs that can't handle it.
- Now **drag the file** to be converted to the *Legacy text file* box



DropTEC

File Edit Help

Mapping file: <drop compiled mapping (.tec) file here>

Legacy text file: <no mapping loaded>

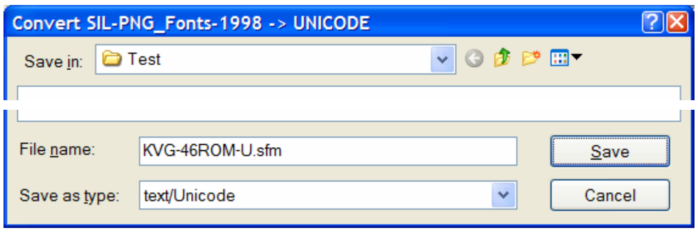
Unicode text file: <no mapping loaded>

Unicode output form: [✓ UTF8] UTF16BE UTF32BE UTF16LE UTF32LE

Normalization: None [✓ NFC] NFD

Generate BOM

- In the dialog box that comes up you can change the folder and/or filename, if you wish
- then click on **Save**
- And voilà, there's your Unicode version of the file!



Convert SIL-PNG_Fonts-1998 -> UNICODE

Save in: Test

File name: KVG-46ROM-U.sfm

Save as type: text/Unicode

By default *DropTEC* adds "-U" to the filename when converting from Legacy to Unicode or "-B" (for Byte) when converting from Unicode to Legacy. You can change the filename before clicking *Save* or renaming the file later if you want.

[7] Text files = Unformatted text only files (as used by Paratext, Shoebox and Toolbox, for example) that can be viewed/edited with a simple text editor like Notepad, in contrast to Word, Excel, Publisher, PageMaker documents, etc.

[8] or any other appropriate .tec file, if it is not for PNG SIL encoded data (so always in this description)

Note: When converting a Paratext project; don't forget to also convert the <*language-name*>**.lds** file, which contains your languages properties, and then adjust the **font** in **Tools – Language Properties and Settings…** of Paratext 6 to the proper Unicode font!

6.2.2 *Using a Batch file with txtconv.exe to convert many single-encoded text files*

Although the above describe approach is very simple and fast for just a few files, it might become tedious and error prone if you have to convert many files, e.g. a whole NT.

With an old fashioned **Batch file** and DOS commands you get this done much faster!

Copy the following lines into a text file in the folder where you have the files to be converted and call it **ConvText.bat** (or whatever other filename you desire, with *.bat* as extension):

`md Unicode`
`FOR %%f IN (*.sfm) DO "txtconv" -nfc -t "C:\Program Files\Common`
`    Files\SIL\MapsTables\SIL-PNG.tec" -u 1 -i "%%f" -o "Unicode\%%f"`
`pause`
	← needs to be
	← on one line!

The above batch file creates a subfolder *Unicode* in the current folder and then runs the **txtconv** program on each file with *.sfm* extension, copying the Unicode output files into the *Unicode* folder.

For running the conversion, **just double-click on ConvText.bat** and you will get a whole New Testament converted to Unicode **in just a few seconds** (try it out!)

Note: The **txtconv** program silently overwrites any existing data, so you might want to move the converted files to a 'safer' place when the conversion is done!

The parameter `-nfc` tells the program to convert to fully composed characters, and `-u 1` means that it should use a replacement character if it finds any unrecognized character and generate a warning (see section Troubleshooting, section 7.2, page 26, if this happens).

You might want or have to adjust the batch file to your needs.

Below is a summary of how to use **txtconv**. This will be displayed when you type 'txtconv' in a command prompt and press *Enter*.

```
>txtconv
Usage: txtconv -i inFile -o outFile [-t tecFile] [-r] [-if inForm] [-of outForm]
 [-nobom] [-nf[cd]] [-u n]
    Required arguments:
        -i <file>   input file
        -o <file>   output file
    Optional arguments:
        -t <file>   compiled TECkit mapping (.tec) file
        -r          reverse (RHS->LHS, or Unicode->Byte) mapping
        -if <form>  input encoding form
        -of <form>  output encoding form
        -nobom      don't write a BOM to Unicode output
        -nf[cd]     apply NFC or NFD normalization to Unicode output
        -u <n>      handling of unmappable input:
                        0 = use replacement character
                        1 = use replacement but generate warning
                        2 = stop conversion
    Encoding forms:
        bytes utf8 utf16be utf16le utf16 utf32be utf32le utf32
```

(If you get an error *'txtconv' is not recognized as an internal or external command …* , when trying to run the batch file or *txtconv* directly, then see Troubleshooting, section7.4, page 26)

Here is a sample batch file ("NFC.bat") that uses txtconv to convert Unicode data to NFC (Composed form):

```
@IF .%1.==.. (
    @echo Normalize Unicode files as Composed:
    @echo Usage: NFC inputfiles
    @echo.
    @pause
) ELSE (
    @echo Normalize Compose %1 files into %1.out
    @FOR %%a IN (%1) DO "TxtConv.exe" -i "%%a" -o "%%a".out -if utf8 -of utf8 -nfc -u 2
)
```

And here is another one ("NFD.bat") that uses txtconv to convert Unicode data to NFD (Decomposed form):

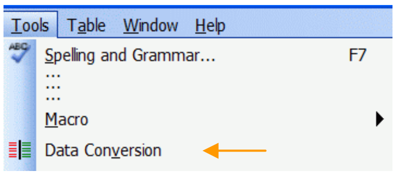
```
@IF .%1.==.. (
    @echo Normalize Unicode files as Decomposed:
    @echo Usage: NFD inputfiles
    @echo.
    @pause
) ELSE (
    @echo Normalize Decompose %1 files into %1.out
    @FOR %%a IN (%1) DO "TxtConv.exe" -i "%%a" -o "%%a".out -if utf8 -of utf8 -nfd -u 2
)
```

6.3 Converting Word documents or files with multiple encodings

Word documents and/or files with **multiple encodings** (e.g. vernacular data plus IPA, Greek, Hebrew or certain European characters) can be converted in Word with **SILConverters**. [9]

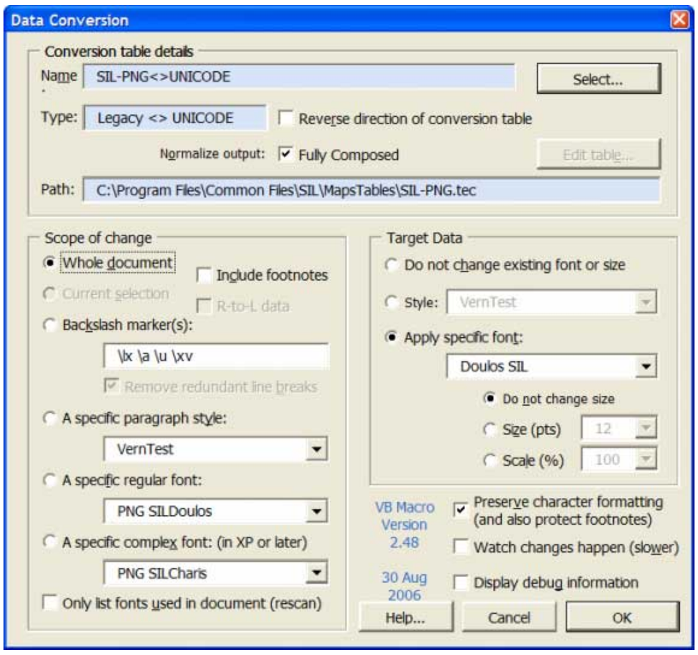
Note: Be sure to make a backup copy of your file before you start!

- To start SILConverters in Word, click on **Tools** – **Data Conversion**.



- In the *Data Conversion* dialog box click on **Select**, to choose the conversion type you want (if it's not already selected)
- Choose the **Normalization** type you want (see explanation in section 5.1, page 16), if it is not already set by default.
- Under **Scope of change** you can now choose to convert the whole document or just certain backslash markers (SFMs), a specific style, or a specific font, depending on your needs.
- Under **Target Data** you can specify if you want to replace the style with another one (already defined with a Unicode font) or if you want to replace the converted data with a certain font.

  The latter is probably what you might want to do in most cases.



Data Conversion

Conversion table details
Name: SIL-PNG<>UNICODE [Select...]
Type: Legacy <> UNICODE ☐ Reverse direction of conversion table
Normalize output: ☑ Fully Composed [Edit table...]
Path: C:\Program Files\Common Files\SIL\MapsTables\SIL-PNG.tec

Scope of change
● Whole document ☐ Include footnotes
○ Current selection ☐ R-to-L data
○ Backslash marker(s): \lx \a \u \xv
☑ Remove redundant line breaks
○ A specific paragraph style: VernTest
○ A specific regular font: PNG SILDoulos
○ A specific complex font: (in XP or later) PNG SILCharis
☐ Only list fonts used in document (rescan)

Target Data
○ Do not change existing font or size
○ Style: VernTest
● Apply specific font: Doulos SIL
● Do not change size
○ Size (pts) 12
○ Scale (%) 100

VB Macro Version 2.48 30 Aug 2006
☑ Preserve character formatting (and also protect footnotes)
☐ Watch changes happen (slower)
☐ Display debug information
[Help...] [Cancel] [OK]

- If you have **several styles or fonts** in Word to convert, run Data Conversion individually for each style (unless you choose *Whole document* for a single-encoded document).
- Similarly, if you have **multiple encodings** in the file (e.g. PNG font, IPA, Greek, etc), run Data Conversion separately for each encoding.

[9] For Toolbox/Shoebox files and sfm files in general, see also other conversion options in section 6.4

- More Information is available by clicking on **Help**.

  Note: Make sure that after converting your Word document to Unicode you also update the fonts in the appropriate style definitions!

  You might want to use the *PNG Unicode Macro* mentioned in section 3.3.2, page 10, to check if everything is converted/updated correctly.

6.4 *Other options for converting multi-encoded SFM files*

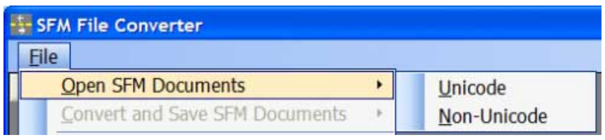
SFM[10] files with multiple encodings (e.g. *Toolbox* files) can also be converted with either the more user friendly Windows program **Bulk SFM Converter** or the command line program **sfconv.exe**.

> **Important**: When converting Shoebox/Toolbox files to Unicode don't forget to also convert your **language properties** (*.lng*) files in the Toolbox Settings folder!
>
> After the conversion, when running Toolbox, make sure that the **correct fonts are chosen** and **Unicode is enabled** in each of the **Language Encoding** entries. To do that click on *Modify* for each of the entries (in Toolbox under: *Project – Language Encodings…*), then choose the *Options* tab, click on *Choose Font…* and select the Unicode font needed. Then click on *Advanced* (still in the Option tab) and make sure that *Unicode (UTF8)* is checked.

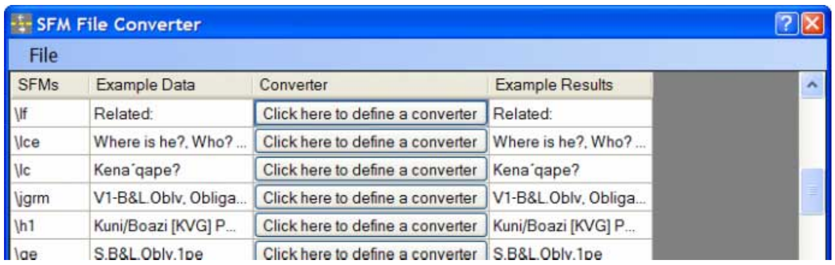
6.4.1 Converting SFM files with Bulk SFM Converter

The **Bulk SFM Converter** (also called **SFM File Converter**) lets you **convert multi-encoded SFM files in one single pass**.

- Start the SFM File Converter program, and open the SFM files to be converted (e.g. Toolbox dictionaries) with **File – Open SFM Document – Non-Unicode**

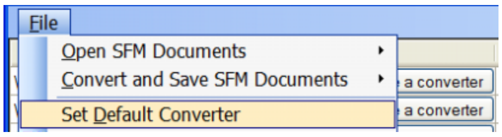


- In the upcoming list of Standard Format Markers you can then **choose the appropriate converter for each SFM.**
- Note: Each of the columns can be easily sorted in ascending or descending order just by clicking on its header.

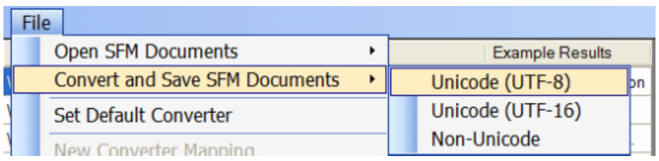


SFMs	Example Data	Converter	Example Results
\lf	Related:	Click here to define a converter	Related:
\lce	Where is he?, Who? ...	Click here to define a converter	Where is he?, Who? ...
\lc	Kena´qape?	Click here to define a converter	Kena´qape?
\lgrm	V1-B&L.Oblv. Obliga...	Click here to define a converter	V1-B&L.Oblv. Obliga...
\h1	Kuni/Boazi [KVG] P...	Click here to define a converter	Kuni/Boazi [KVG] P...
\oe	S.B&L.Oblv.1pe	Click here to define a converter	S.B&L.Oblv.1pe

- Before you start choosing the appropriate converter for each SFM, it might be best to define a **default converter** for the conversion type you will use for most SFMs. This will save you a lot of work!

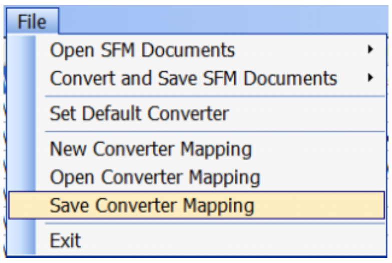


- When all the setup is done, choose **Convert and Save SFM Documents – Unicode (UTF-8)** and your conversion should be done



- You might want to **save the Converter Mapping** you created in case you ever need it again!

- It can then later be opened again with **Open Converter Mapping**.



[10] SFM = Standard Format Marker, used in applications like Toolbox and Paratext, e.g. \v for verse, \lx for lexeme, etc.

6.4.2 Converting SFM files with sfconv.exe

sfconv.exe is part of the *TECkit Utilities* optionally installed with SILConverters. (See section 4.1 above)

Its parameters are as follows:

```
>sfconv
8-bit to Unicode:
    SFconv -8u [-utf8|-be|-le] [-bom] -c ControlFile [-d MappingDirectory] -i InFile -o
    OutFile
Unicode to 8-bit:
    SFconv -u8 [-utf8|-be|-le]        -c ControlFile [-d MappingDirectory] -i InFile -o
    OutFile
```

Like the *Bulk SFM Converter*, this program allows you to convert sfm files with **multiple encodings** in **one single pass**.

For this to work, you must setup a control file that tells the program what conversion mappings need to be used for the different SFMs.

In the example below (which could be called *SHConvert.xml*)

- the marker *\ph* (using SIL IPA93 Font encoding) will be converted with *silipa93.tec*,
- the marker *\bg* (using SIL Greek Encoding (Galatia)) will be converted with *SILGreek2004-04-27.tec*
- the markers *\lx*, *\lv*, *\1p*, etc (using SIL PNG encoding) will be converted with *SIL-PNG.tec*
- and all the rest (as defined in *defaultMapping*) using the “standard” (English) Windows encoding, will be converted with *iso-8859.tec*.

```
<?xml version="1.0"?>
<!--Convert Shoebox Databases to Unicode-->
<sfConversion defaultMapping="iso-8859.tec">
    <sfMarkers escape="\"
                chars="abcdefghijklmnopqrstuvwxyz_ABCDEFGHIJKLMNOPQRSTUVWXYZ^#.-
    0123456789!*"
                mapping="ISO-8859-1">
        <marker name="ph" mapping="silipa93.tec"/> <!-- Lang=IPA -->
        <marker name="lx" mapping="SIL-PNG.tec"/> <!-- Lang=Kuni -->
        <marker name="lv" mapping="SIL-PNG.tec"/> <!-- Lang=Kuni -->
        <marker name="1p" mapping="SIL-PNG.tec"/> <!-- Lang=Kuni -->
        <marker name="1pe" mapping="SIL-PNG.tec"/> <!-- Lang=Kuni -->
        <marker name="1pi" mapping="SIL-PNG.tec"/> <!-- Lang=Kuni -->
        <marker name="1s" mapping="SIL-PNG.tec"/> <!-- Lang=Kuni -->
        <marker name="2p" mapping="SIL-PNG.tec"/> <!-- Lang=Kuni -->
        <marker name="2s" mapping="SIL-PNG.tec"/> <!-- Lang=Kuni -->
        <marker name="3p" mapping="SIL-PNG.tec"/> <!-- Lang=Kuni -->
        <marker name="3s" mapping="SIL-PNG.tec"/> <!-- Lang=Kuni -->
        <marker name="bg" mapping="SILGreek2004-04-27.tec"/> <!-- Lang=Greek -->
    </sfMarkers>
</sfConversion>
```

Depending on the complexity of the file(s) to convert, you might end up with many dozens of such entries.

You can find more information about **sfconv** in the chapter *Converting Standard Format files: SFconv* of **TECkit_version_2.1.doc.pdf**, which you will find in **Start Menu: SILConverters – TECkit – Help – TECkit version 2.1** or directly in the folder **C:\Program Files\Common Files\SIL\Help**.

6.4.3 Converting Shoebox/Toolbox files with SH2SH

Still another option for converting Shoebox/Toolbox files is the program **SH2SH**, which is part of **the Shoebox Utilities** available at http://scripts.sil.org/SHUtils-manual.

The website says about it: *sh2sh … outputs a Unicode Shoebox database file. Thus it converts all fields too or from Unicode according to the encoding information in the language properties.*

This might be the easiest ways to convert Shoebox/Toolbox files as it takes the encoding information for all SFMs from the language properties. Therefore you don’t have to list all SFMs and specify how to convert them. This is done automatically.

(Unfortunately an earlier version of this program didn’t quite do what was expected and a newer version has not been tested.)

6.5 Converting PageMaker files

PageMaker is not Unicode compatible, so you will have to move your data to other applications.[11]

To do this

- you can either cut and paste your data from PageMaker to Word (which might be rather tedious, because you need to do this for each individual frame) OR
- you can export the whole PageMaker document as HTML, then open the resulting file in Word. Unfortunately the layout of the outcome is not very accurate, but you have at least saved your data!

In Word you might then have to change the font to your old PNG font first (as PageMaker's export function seems to forget what font you used there) and then run SILConverters.

After that, make any necessary changes to the layout in Word (or copy the data to Publisher, or Scribus[12], or whatever application you wish, and do the layout there)

6.6 Converting Publisher files

Unfortunately there is currently no tool available that lets you directly convert Publisher files to Unicode.[13]

You can, however, use Publisher's *Edit Story in Microsoft Word* feature (under *Edit*) and run SILConverters there. Quitting Word then takes you back to Publisher and you will have your data converted to Unicode.

Unfortunately you will need to do that separately for each individual text box.

Be sure to make a backup copy of your file before you start!

6.7 Converting Excel files

There is currently no tool available that lets you directly convert Excel files to Unicode.

Depending on the complexity of your file and whether it uses just one or several encodings, you can either cut and paste a whole worksheet or individual data (e.g. certain columns) to Word and run the Data Conversion procedure there, then paste the outcome back to Excel.

After re-importing the data you might have to make adjustments to the fonts being used and some column/row borders.

Unfortunately attempts to save an excel file as *.xml*, *.csv*, *.txt*, etc., then convert it to Unicode, and re-import it into Excel have not been successful, as in most cases Excel does not correctly export 'hacked' fonts.

Exporting to *.html*, converting it, and re-importing into Excel might (or might not) work, depending on the complexity of your spreadsheet and the specific characters you use.

Be sure to make a backup copy of your file before you start!

6.8 Converting Access databases

For Access databases pretty much the same applies as for Excel files, except that you have to cut and paste your data from individual Access tables instead from worksheets.

[11] *InDesign*, the successor of *PageMaker*, is Unicode compatible, but you will (most probably) still have to export your data from *InDesign*, convert it to Unicode and re-import it.

[12] *Scribus* is a free Open Source Desktop Publishing application

[13] SIL's Unicode Transition Initiative might actually be willing to come up with some tools if there is a real need.

7 Troubleshooting

7.1 Accent marks (diacritics) in Unicode fonts are not displayed properly

If diacritics on certain characters in Unicode fonts are not displayed properly, e.g. if they are too low/high or shifted to the side, this might have the following causes:

7.1.1 Font

The **font** you use might have an error or just not be fully Unicode compliant. Check if you really use the newest version of it!

7.1.2 Uniscribe (usp10.dll)

The Microsoft **Uniscribe** rendering system (**usp10.dll**), that most programs use for rendering Unicode fonts, might be outdated and/or faulty.[14]

You might find copies of usp10.dll in:

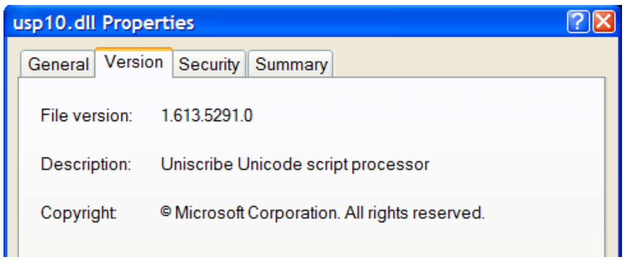
- *C:\WINDOWS\system32* (and maybe also in C:\WINDOWS)
- *C:\Program Files\Common Files\Microsoft Shared\OFFICE11* (or somewhere similar depending on your MS Office version
- *C:\Program Files\Paratext 6*

If the rendering problems occur in an MS Office program or Paratext 6, try replacing the usp10.dll in their respective folders mentioned above.

To find out what version of Uniscribe you have, right-click on the file, choose *Properties* and click on the *Version* tab.

You should find a new version in the Unicode section on the LCORE Website and/or in the relevant subfolder on the SIL PNG FTP site. One of the newer versions available (newer than Office 2003 and Paratext 6) is 1.613.5291.0.

Still newer versions come with Windows Vista and Office 2007 (but unfortunately they have some bugs too!).



File version: 1.613.5291.0

Description: Uniscribe Unicode script processor

Copyright: © Microsoft Corporation. All rights reserved.

Unfortunately, in Windows XP replacing the usp10.dll in *C:\WINDOWS\system32* is not quite as easy, because if you replace or delete it, the operating system will automatically restore the original! Therefore you will need to copy the file there when Windows is **not** running.

> **Note:** NRSI no longer recommends replacing the system usp10.dll in *C:\WINDOWS\system32*. If you need a newer version of it for a specific application, just add usp10.dll to the folder where the .exe file of that application resides. This should take care of the problem and is much less bothersome.

But if you really want or need to replace the system usp10.dll you can do this as follows:

- Copy the version you want to install to *C:\usp10.dll*, for example.
- Shut down Windows and boot from the original Windows XP installation CD (or another bootable CD if you prefer)
- With the Windows XP installation CD, when the message "What would you like to do?" comes up, enter "R" for *Repair.* Later, when a message `1) Windows` comes up, press *1* and *Enter*.
- After that type the following commands, pressing *Enter* after each of them

```
cd \windows\system32
copy usp10.dll usp10.old
copy c:\usp10.dll c:\windows\system32\
exit
```

- Now, when you start Windows again, the new usp10.dll will be installed.

[14] An alternative to Uniscribe is SIL's **Graphite** rendering system, which is used in FieldWorks and WorldPad (and some versions of OpenOffice.org and Mozilla programs). In contrast to Uniscribe, Graphite (at least in FieldWorks and WorldPad) allows the user to choose variant glyphs (at least in FieldWorks and WorldPad), e.g. different sizes of a Glottal stop character.

7.2 Faulty Conversion output

If after conversion to Unicode (or back to Legacy) your output does not look as expected and/or if the procedure warns you about unrecognized characters, check the following:

- Do you display your data with the **correct font** and is it the **newest version**?
- Did you use the **correct conversion table/mapping** (e.g. SIL-PNG<>UNICODE, SIL-IPA-1990<>UNICODE, SIL PUA 3.2<>UNICODE 5.0, etc.) for your conversion?
- Did you use the **correct conversion direction** (Legacy to Unicode, vs. Unicode to Legacy)?

7.3 Error in TECkit Map Unicode Editor

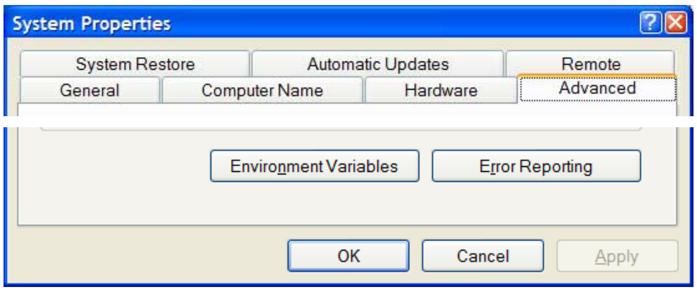
If you get an "**unexpected character: "?" at line 1**" Error in the *TECkit Map Unicode Editor*, you probably have an older version of the *TECkit* DLLs (i.e. *TECkit_Compiler_x86.dll* and *TECkit_x86.dll*) somewhere on your computer, either in *C:\WINDOWS\system32* or elsewhere referenced by your system path (probably installed by another program, e.g. Paratext or LinguaLinks).

To overcome this problem, copy the newer versions of **TECkit_Compiler_x86.dll** and **TECkit_x86.dll** from *C:\Program Files\Common Files\SIL* to *C:\WINDOWS\system32*. (Simply deleting these files or deactivating them by renaming their extension doesn't work, as Paratext insists to have these files there and will just recopy the old versions there again.)

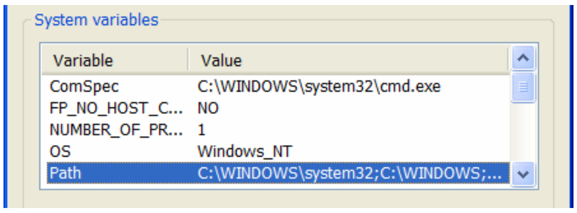
7.4 '…' is not recognized as an internal or external command

If you get an error message like **'txtconv' is not recognized as an internal or external command, operable program or batch file**, e.g. when running a batch file or running *txtconv.exe*, *sfconv.exe*, etc. in a command prompt window, then there is most probably an error in the **search path** for executable files.

- To check and/or change this path, open *System Properties* (e.g. via the *Control panel* or by right-clicking on *My Computer* and choosing *Properties*),
- Then choose the **Advanced** tab, and click on **Environment Variables**



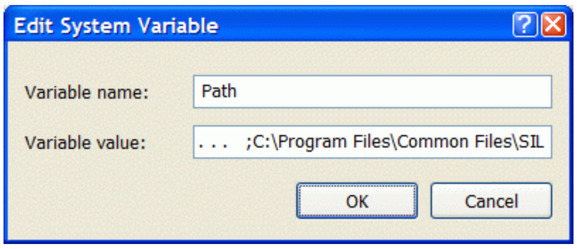
- In the Environment Variables dialog box, double click on the **Path** entry (under System Variables)



Variable	Value
ComSpec	C:\WINDOWS\system32\cmd.exe
FP_NO_HOST_C...	NO
NUMBER_OF_PR...	1
OS	Windows_NT
Path	C:\WINDOWS\system32;C:\WINDOWS;...

- In **Edit System Variable** make sure *C:\Program Files\Common Files\SIL* is part of the path and all the rest of the path is correct too (e.g. there must be a semicolon ";" between each path entry). You might want to copy and paste the entire path to a text editor for easier examination and paste the corrected entry back again.

  Note: Be careful not to mess up this path as otherwise other programs depending on it might not work correctly any more!



Variable name: Path

Variable value: . . . ;C:\Program Files\Common Files\SIL