The TLex Suite

2002 - 2026

Professional Lexicography, Terminology and Corpus Query Software

TLex/tlTerm User Guide

Version 15.0.31

David Joffe
&
Prof. Dr. Gilles-Maurice de Schryver

Copyright © 2004-2026 TshwaneDJe Human Language Technology
https://TshwaneDJe.com/ djsoftware.tech

Table of Contents

General Notes to the User Guide....1
Terminology Used in this User Guide....1
Frequently Asked Questions (FAQ)....1
ChangeLog....1
Additional Resources....1
Free Tutorial Videos....1
Training; Support; Customisations....2
Getting Started with TLex: Quick Guide....3
Creating a New Dictionary Project....3
Overview of the TLex/tlTerm Interface....4
Monolingual Editing Environment / tlTerm....5
Bilingual Editing Environment....6
Entry List [1] ('Lemma List' in TLex, and 'Term List' in tlTerm)....6
QuickSearch Box....6
Tree View [2]....7
Attributes (F1, F2) and Tools (F3, F4, F5, F6) Window [3]....7
Preview Area [4]....7
Bilingual "Side-by-Side" View Mode....7
"Expanded View" Mode vs. Side-by-Side View Mode....8
Adding New Entries to the Dictionary/Termbase....8
Structuring Articles with the Tree View Control....8
Changing the Order of Elements....9
Changing the Order of Homonyms....9
Marking Entries as 'Incomplete'....9
Saving Your Work....9
Restoring the Last-saved Version of an Entry ('Undo')....10
Adding a Batch of New Entries, Or 'Import Wordlist/CSV'....10
Deleting Entries from the Dictionary/Termbase....10
Printing your Dictionary....10
Making Regular Backups....10
Manual Backups....10
Automatic Backups....11
Offsite Backups....11
"Cloud" Backups....11
Getting Started with tlTerm: QuickStart Guide....12
Creating a New Termbase....12
Primary Differences Between tlTerm and TLex....13
Terminology is Concept-Centric....13
Adding Synonym Cross-References in tlTerm vs TLex....14
Adding Custom Fields....14
📷 Snapshots and "Show Differences" (Review Changes)....16
Introduction to Snapshots....16
More Advanced Snapshots Functionality....16
Filtering to Show Entries With Differences (Review Differences to Multiple Entries)....16
[Advanced] Creating and Saving Sets of Snapshots....17
Changing the Interface Language (Localisation)....18
Built-in Localisation Editor – Creating New Interface Languages....18
Keeping Your Software Up To Date....20
"Wide View" Mode and "High View" Mode....21

F12 Overlay Window (Larger Attribute Editing Window)....22
Attributes (F1) and (F2)....23
Attributes (F1)....23
"Incomplete" Checkbox....24
Attributes (F2) – Closed Lists with Multiple Selection....24
Editing Attributes....25
Expansion of Tilde Character (~)....25
Overriding the Tilde Substitution Text....25
Using Text Formatting Within a Text Field....26
Markup Characters (Bold, Italics, Underline etc.)....26
Smart Formatting Reversal (%b, %i vs. %B, %I - "soft" vs. "hard" bold/italics)....27
Using Inline Elements for Text Formatting [Advanced]....27
Configurable Keyboard Shortcuts....27
Inline Elements: Special 'Tagging' Shortcut Keys [Advanced]....29
Replace As You Type....29
Configuring Replace-As-You-Type [Advanced]....29
Inserting Phonetic Symbols (e.g. IPA)....29
Editing TLex "Smart Cross-references"....31
"Smart" Cross-references: Overview....31
Adding Cross-references....32
Editing Cross-references....33
Customising the Cross-reference Types....33
Allowing Cross-references Between Sections in Multi-section Documents....33
Search (F3)....34
Overview....34
Search Options....34
"Case Sensitive"....34
"Whole Word Only"....34
"Regular Expression" Searches [Advanced]....34
A Few Useful Regular Expression Example Applications....36
"Find" Tool....36
"Search and Replace" Tool....37
Searching F2 Lists with "Find" and "Search and Replace"....37
Search "Smart Presets"....37
Format (F4)....39
Alternative Sets of Labels for Lists....39
Cross-reference Label Sets....39
Show Related Cross-references [TLex]....40
Expand Entities....40
Show Frequency....40
Expand Tildes (~)....40
Show Timestamp....40
Show Usernames....40
Find/Filter (F5)....41
Types of Filter Conditions....41
"Reveal" Filter....42
"Hide" Filter....43
Combining "Reveal" and "Hide"....43
Extracting or Printing a Subset of the Dictionary/Termbase....44
Corpus (F6)....45
Configuring the Corpus....45

Doing a Corpus Search....45
Sorting the Results....45
Auto-grabbing Usage Examples....45
Copying Selected Examples (Corpus Lines) to the Clipboard....46
Corpus Encryption....46
Corpus Registry Settings [Advanced]....46
Increasing the Maximum Number of Corpus Lines Returned....46
Increasing the Maximum Number of Cached Sets of Search Results....46

Tags....48
Overview....48
Tagging Commands....48
Tag Filter....49
Tagging in the “Find” Tool....49
Tagging a Range of Entries (e.g. an Alphabetic Section)....49

Microsoft Word/Excel Integration (‘Search Widget’)....50
Microsoft Word/Excel ‘Search Widget’....50
Sending/Adding Text from TLex/tlTerm to Word/Excel....51
General 'Text Commands' Menu....51
General Text Commands Menu in tlCorpus, and Time-saving Linked Mode....51

TLex Bilingual Editing Features....52
Linked View....52
Automatic Lemma Reversal....52
Single Article Reversal....52
Full Dictionary Reversal....52
Translation Equivalent / Term Fanouts [TLex & tlTerm]....53

Compare/Merge....54
Overview....54
Action Commands: “Add”, “Merge”, “Replace” and “Delete Left”....55
Settings....56
Bilingual Dictionaries: Choosing Which Sides to Compare....56
Batch Merge....56
[Advanced] Comparing DTDs (Document Type Definition / Field Descriptors)....57
Merging DTDs....57
Merging User Lists....58

Ruler Tool [Advanced]....59
What is a Multidimensional Lexicographic Ruler?....59
Rulers in TLex....59

Customising the Dictionary Grammar or Termbase Fields using the DTD [Advanced]....62
Basics of Hierarchical Data Modelling....62
Elements and Attributes....62
What is a DTD?....63
Element Types....63
Element Attributes....64
Element Child Relations....65
Element Child Relation Constraints....66
Synchronizing the Child Element Order with the Output Order in the Styles....67
Copy Child Element List to Clipboard....68
Special Element Types (TLex-internal)....68
Dictionary and Language Elements....68
Lemma Element....68
References Element....68
The TLex Default DTD....69

Elements....69
Attributes....69
Attribute Lists....71
Multimedia (Audio and Images)....75
Special Element and Attribute Types (User-configurable)....75
Translation Equivalent (TE)....75
Frequency....76
Sense....76
Display Lemma Sign....76
Internal 'Notes'....76
DTD Templates....77
Customising the Cross-reference Types....78
Renaming, Removing and Adding Cross-reference Types....78
Changing the Output Order of Cross-reference Types....78
Creating Singular and Plural Labels for Cross-reference Types....78
Customising the Language used to Display Cross-reference Types....79
Customising the Alphabetic Sorting....81
Overview....81
Table-based Sorting....81
Configuring Table-based Sorting....81
Selecting Other Sort Plug-ins....82
Loading/Saving Sort Configurations....83
Styles System....84
Overview....84
Styles/formatting....84
Basic Formatting Options....84
Font Names Starting With "@"....85
Colours....85
Text Before/After Options....86
Group Style Properties for Multiple Elements or List Items....86
Automatic Numbering....87
Paragraph Style Options ("Indents and Spacing" Tab)....89
Using Paragraph vs "%n" to Generate New-lines Before/Between Elements....89
Paragraphs and "Show whitespace" Option....89
Paragraph Margins (Spacing)....90
Creating a Hanging Indent....90
Creating Nested Indents....90
[Advanced] Notes on Using Paragraph Styles....90
Output (Display) Order....91
Element and Attribute Output (Display) Order....91
"Visible" Flag (Non-Printing Fields)....92
Entities....92
Using Entities to Embed Labels within Other Fields and Further Customisation....93
Style Sets, Or 'One Database, Many Dictionaries'....95
"Masks" (Selectively Hiding Specific Information on a Per-Edition Basis)....96
Configuring Masks....97
"Smart Styles" (Dynamically Customisable Styles) [Advanced]....99
Creating a Thesaurus....100
Multi-Document Smart-Search....101
Importing Data into TLex/tlTerm....102
Import Wordlist / CSV (Comma Separated Values)....102
Character Encoding Issues....103

Importing XML [Advanced]....104
'Merge' XML Import....105
Post-Import Processing / Data Remodelling....105
“XML Line by Line” Importer....105
tlTerm: Importing Terminology from Other Formats....107
CSV (Comma-Separated Values) or Excel....107
TBX (TermBase eXchange) Format....107
TMX/XLIFF (Batch Import using Multi-Document Tool)....107
Exporting Data from TLex/tlTerm....109
Copying Lemma Signs, Terms and Articles....109
Copy Lemma Sign / Term....109
Copy Entry Text....109
Copy Entry HTML....109
Exporting the Database, in Part or in Full....109
Lemma Signs [TLex] / List of Terms [tlTerm]....109
Text....110
RTF (Rich Text Format)....110
HTML (Web Page)....111
XML (eXtensible Markup Language)....111
XML (Formatted)....112
Exporting to InDesign....112
Exporting to InDesign via RTF....112
Exporting a Range of Entries or a Single Alphabetic Section, e.g. “A”....113
Network (ODBC Database) Support [Advanced]....114
Overview....114
Troubleshooting....114
Configuring an ODBC Database....115
Configuring macOS MySQL Client (64-bit Sierra, Mojave, Catalina etc.)....115
Important Note for 64-bit Windows Clients....116
“Cached” ODBC (ODBC, Sped Up)....117
Entry Locking....117
Locking the Database....117
Optimisation Tips....118
Notes for PostgreSQL....118
Timezone Settings....118
Slow Performance (PostgreSQL)....118
Commandline Options [Advanced]....119
Automatically Logging On to the Document Passed on the Commandline....119
Watch Folders [Advanced]....120
Lua Scripting [Advanced]....121
Getting Started with Lua Scripting in TLex/tlTerm or tlDatabase....121
Lua Script Attributes....121
Stand-alone Scripts....122
Stand-alone Entry Scripts....122
API Reference Documentation, Tips, Samples and Further Information....122
Other Uses of Lua Within TLex/tlTerm....122
“Smart Styles”....122
Lua Filters....123
Lua Sorting....123
"Click Scripts" And "Search Scripts"....123
User Management....124

Configuring User Logins....124
Deleting Users: “Delete” vs. “Purge”....124
Privileges System....124
Field-Specific Privileges....125
Monitoring and Tracking Progress....125
User Progress Statistics....125
XSLT Export Transforms (Extensible Stylesheet Language Transformations)....126
Initial Configuration....126
Multi-pass (Multi-stage) Transforms....127
tlTerm and TLex Online Publishing system....128
Introduction....128
[Advanced] Technical Details....128
“Best Practice” Configuration and Workflow....128
Other Optional Settings:....130
BatchSize Setting and Publish Speed....130
[Advanced] Generating SQL Command Files without actual publish....130
[Advanced] Working with Multiple Online Publishing Profiles....131
[Advanced] Publishing to Your Own Custom Online Platform....131
Troubleshooting....131
Duplicate Entries....131
[Programmers] MySQL Publish Database Schema....132
[Advanced/Programmers] Notes on the meaning of the “scores”....132
[Advanced/Programmers] Notes on Entry IDs....132
tlCorpus (Time-saving) Linked Mode....133
Starting tlCorpus in Linked Mode....133
Troubleshooting Linked Mode....134
[Programmers] Linked Mode “Web API”....134
Multi-section Documents....136
Introduction....136
[Intermediate] Adding, Renaming or Deleting Sections....136
OpenAI Integration (GPT) [v2023+ version only]....138
Feature Availability: v2023+ Version only....138
Using OpenAI with TLex (and/or tlTerm, tlCorpus, tlDatabase etc.)....138
How to get an OpenAI key?....139
How to enter your OpenAI key in TLex Lexicography Software?....139
Using the Service in TLex, tlTerm, tlCorpus etc....140
OpenAI TLex/tlTerm/tlCorpus Settings....140
Automated and Semi-Automated Entry Creation....140
Word and Excel Integration with AI integration....141
Multi-Document Search and AI integration....141
Custom User Prompts....141
Query Rate Limits and Timing....142
Setting Up LM Studio or ollama (litellm) AI Servers....143
Feature Availability: v2023+ (v15+) Version only....143
Setting Up LM Studio AI or litellm ollama AI Servers in TLex/tlTerm/tlDatabase....143
Step 1: Accessing AI Server Settings....143
Step 2: Configuring AI Servers....143
Step 3: Utilizing AI in TLex/tlTerm/tlDatabase....143
Ollama setup....143
LM Studio Setup....144

General Notes to the User Guide

TLex (also known as TshwaneLex) ("Lex" for "lexicography") is a specialised software application for compiling *dictionaries*. tlTerm (previously called TshwaneTerm) (for "terminology") is a termbase editing application, i.e. a specialised application for compiling and managing terminology lists. (For a brief explanation of the primary differences between TLex and tlTerm, and when to use which, see the 'Getting Started with tlTerm' chapter). tlTranslate is a 'Translation Memory' application, for computer-assisted translation. The name 'Tshwane' stems only from the traditional African name for Pretoria, the city where the software was originally developed - the software itself, however, is fully internationalised, and can be used for virtually any language in the world. All fields in TLex, tlTerm, tlCorpus and tlTranslate will thus accept **Unicode** characters (i.e. characters from any language). If you have problems displaying characters from a particular language, this is usually just a configuration problem (e.g. choosing the correct font).

Terminology Used in this User Guide

Throughout this document, the terms "**lemma**", "**entry**" and "**article**" are generally used interchangeably or near interchangeably. The terms "**lemma sign**" and "**headword**" are used nearly interchangeably. The phrase "**translation equivalent**" is always used to refer to a more or less equivalent phrase in the *target* language in a bilingual dictionary, while the term "**definition**" is always used to refer to a monolingual *explanation* in the *same language* as the headword. The terms "**element**" and "**attribute**" stem from the terminology of the XML industry standard, on which TLex, tlTerm and tlTranslate are based. The terms "**subentry**" and "**combination**" are also used interchangeably.

The terms "**termbase**", "**terminology database**" and "**terminology list**" are used largely interchangeably.

Frequently Asked Questions (FAQ)

For additional documentation, tips and tricks, answers to common queries, information on undocumented features, and other information for getting the most out of your software, you should also regularly check out the FAQ ("Frequently Asked Questions") document, available online at: http://tshwanedje.com/faq.html. The FAQ also contains supplementary documentation (e.g. on new features) that has not yet made it into the User Guide.

ChangeLog

The most up-to-date information on the newest features and fixes in the software is always published in the official ChangeLog at: http://tshwanedje.com/tshwanelex/changelog.html

Additional Resources

Free Tutorial Videos

Free tutorial videos are available online via: http://tshwanedje.com/videos/

Also subscribe to our YouTube channel, where we periodically release new ‘tips and tricks’ videos, how-to videos and demo videos:

https://www.youtube.com/channel/UCD_mfcdc234peP84Ooahgsg

Tutorial/How-to Playlists:

Playlist: Tutorial/How-to Videos (All)

Playlist: tlTerm Tutorial Videos

Playlist: New Users: Introduction to TLex, Getting Started

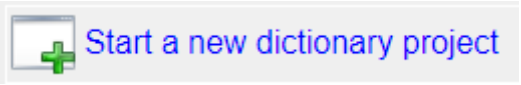
Playlist: Software Tips

Training; Support; Customisations

Paid training is also available from TshwaneDJe - Zoom/Skype-based or on-site. If you require new features, or modifications or extensions to existing features, this is also available as a paid service.

Creating a New Dictionary Project

To start a new dictionary project, open TLex and select the "**File/New dictionary**" menu option, or click "Start a new dictionary project" on the 'Start page':



A dialog will appear prompting for basic information about the dictionary project, such as the name of the dictionary, an optional description, copyright information and URL (Uniform Resource Locator, i.e. website address, if applicable for the dictionary project). Note that these properties may be modified at any later time by using the "Dictionary/Properties" menu option.

Under "Dictionary type", you must select whether you are creating a monolingual or bilingual (i.e. consisting of two "languages/sides") dictionary database. If you want to create a semi-bilingual dictionary, choose the "Bilingual" option. Below this, under "Primary language", enter the name of the main (primary) language for the dictionary (for a bilingual, this will be the "left" half or section), and if creating a bilingual dictionary, the target language name under "Target language" (this will be the "right" half or section). (Note that the "Noun classes" option enables features that are relevant to the Bantu language family.) The "Quicklinks" field is not important to the database itself – this is only used to configure the alphabetic shortcut links that will appear at the top of the TLex Preview Area.

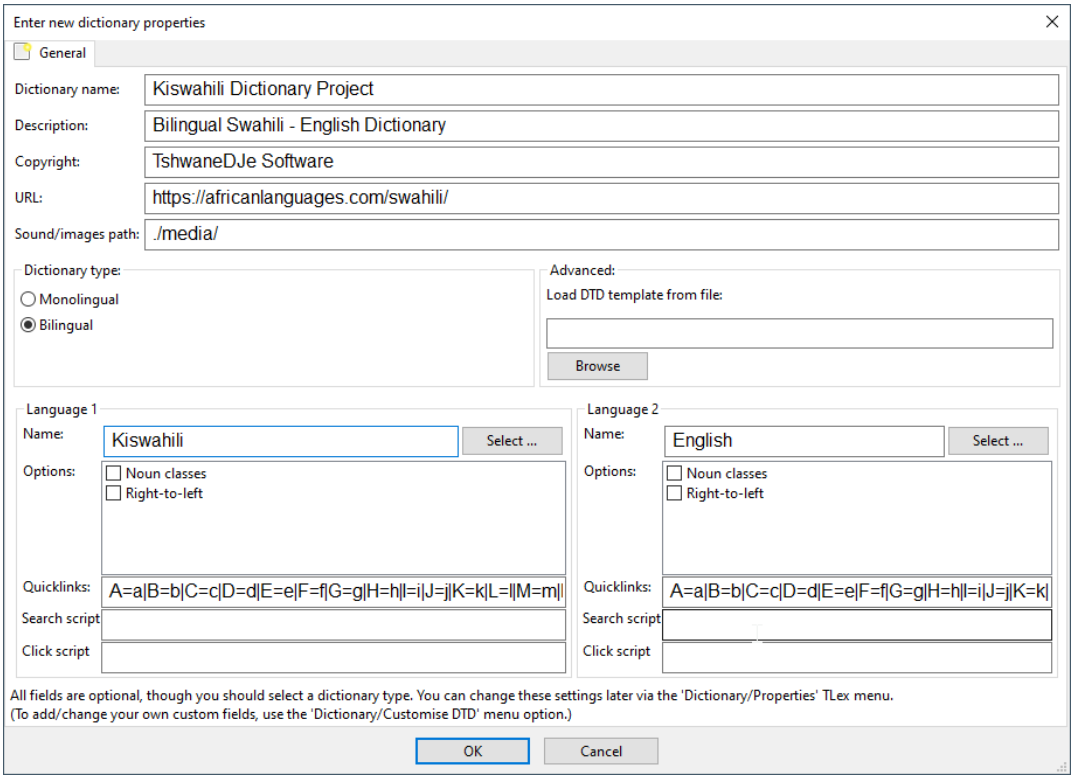


Figure 1: An example of the "New dictionary" dialog for a bilingual Kiswahili – English dictionary project. (You can return to this dialog again later via "Dictionary/Properties" menu.)

Note that all fields in this dialog, as everywhere else in TLex, will accept **Unicode** characters.

> **Tip:** All fields in this dialog are optional, though you should select the '**Dictionary type**' - this is important as it will determine if you have only one "side" (section), or two or more sections (though you can start with one side, and add more later).
>
> The "Sound/images path" is important to set correctly if you want to use sound/images. Typically this will be a path relative to your TLex or tlTerm file - see the included Samples.

Once you are satisfied with the basic properties for your dictionary, click "OK". The TLex editing environment will appear, allowing you to begin compiling your dictionary. Before you proceed, select "File/Save" from the menu (or press Ctrl+S), and enter a filename under which to save your dictionary project. Make sure to remember where you save it. Thereafter, this dictionary database can be opened again by selecting the "File/Open: TLex (TshwaneLex) file" menu option and selecting this file.

Also note that all recent dictionaries you have worked on are listed under the "File/Open recent" menu option. These actions are also available on the "start page" ("Tasks / Open recent") that appears in the centre of the window when you open TLex.

> **Tip:** You can return to the above 'Dictionary properties' dialog later again at any time by using the "Dictionary/Properties" menu command

> This playlist may also be a useful introduction to the basics of TLex:
> https://www.youtube.com/watch?v=8Y2xAt7ox7g&list=PLE2A6074104252830

Overview of the TLex/tlTerm Interface

Once you have created a new dictionary, you will be presented with the main editing environment. The default layout of this will depend on whether you are working on a monolingual dictionary or a bilingual dictionary.

Monolingual Editing Environment / tlTerm

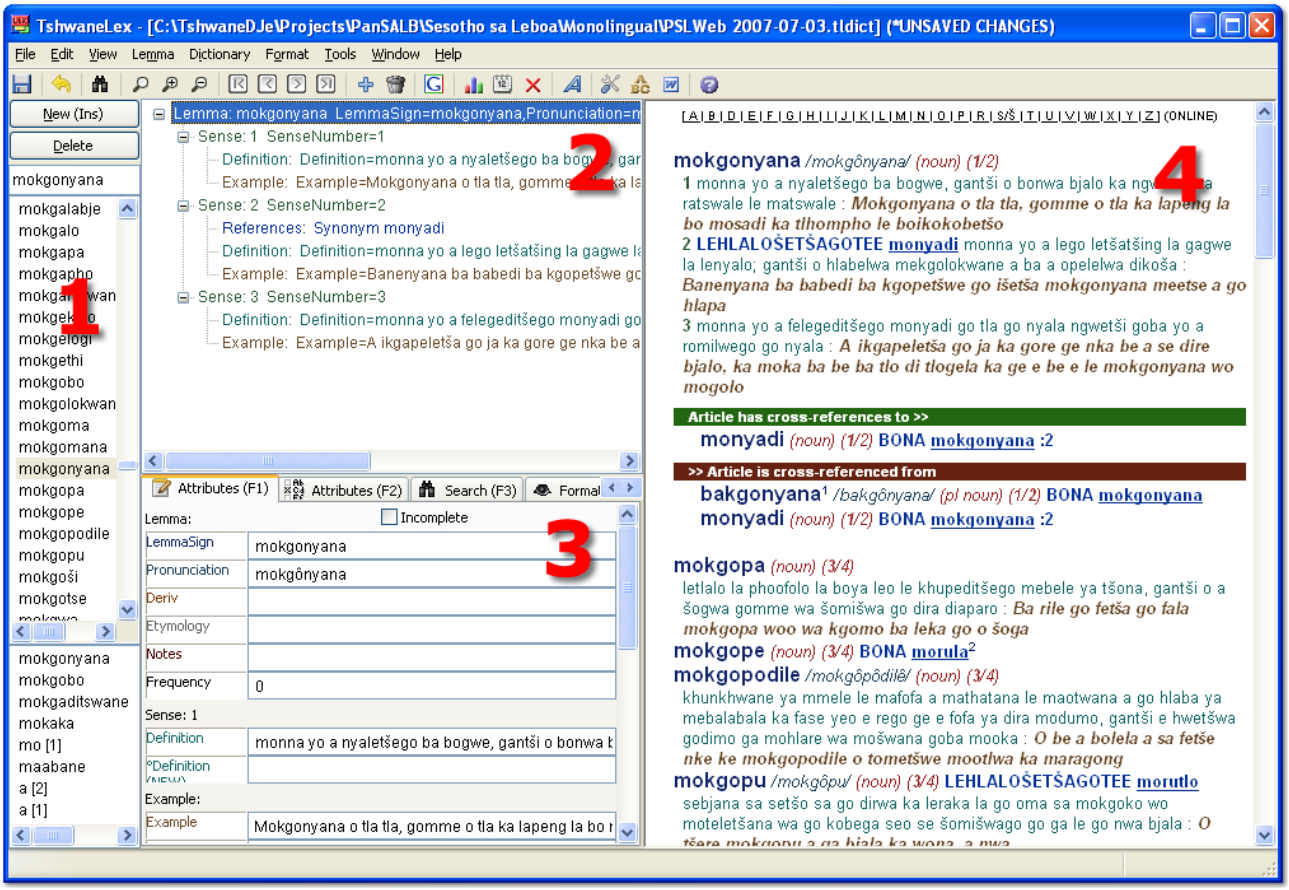


Figure 2: An example of the monolingual editing environment for an explanatory Northern Sotho dictionary [Data online at: http://africanlanguages.com/psl/]

On the screenshot, the four main editing areas have been marked “1” to “4”. These are discussed below.

Bilingual Editing Environment

For a bilingual dictionary, the interface is split down the centre, and similar editing windows for each side of the dictionary are shown side by side.

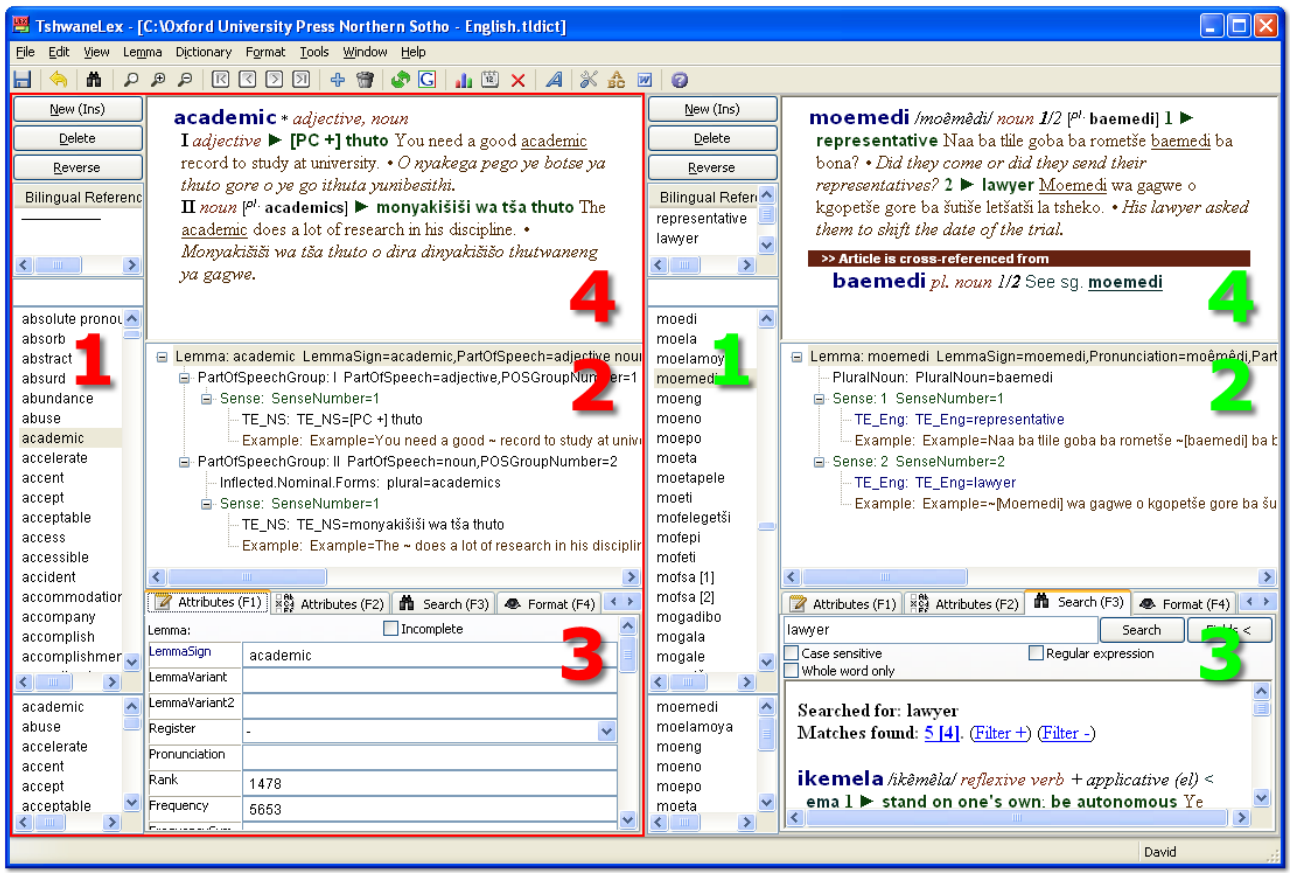


Figure 3: The TLex bilingual editing environment in "Side-by-Side View" mode for the 'Oxford Bilingual School Dictionary: Northern Sotho - English'

The four main editing areas, as indicated by the numbering on the above screenshots, are the "Lemma List" [1], the "Tree View" [2], the "Attributes and Tools window" [3], and the "Preview Area" [4]. Each will now be discussed briefly.

Entry List [1] ('Lemma List' in TLex, and 'Term List' in tlTerm)

This is a scrollable list of all entries in the dictionary/termbase. To view or work on an entry, click on the word / term in this list.

QuickSearch Box

Just above the Entry List is a "quick-search" box. When you start typing a word into this box, TLex/tlTerm will automatically jump to the closest matching entry in the list.

> **Tip:** NB: The **shortcut key "Esc" (escape)** will always immediately take you to the quick-search box from almost anywhere in TLex/tlTerm.

Tree View [2]

Each dictionary article has a hierarchical structure, e.g. a lemma contains senses which may further contain subsenses. The Tree View is used to view or modify this hierarchical structure of an article, e.g. to add senses, subsenses, usage examples, collocations, cross-references, etc. Each node in the Tree View is called an **element**. Right-clicking on any element in the Tree View displays a menu with a list of editing options available for that element, such as adding child elements.

Attributes (F1, F2) and Tools (F3, F4, F5, F6) Window [3]

This window consists of five sub-windows, which can be accessed quickly using the shortcut keys "F1" to "F6". The first two sub-windows are used to edit the so-called **attributes** of the currently selected element in the Tree View, i.e. the actual values that are associated with the element. Thus the Tree View, which shows *elements*, is basically used to modify the *skeletal structure* of an article (e.g. adding a usage example at a certain point), while *attributes* flesh out that structure with actual *content* (e.g. the text of the usage example itself plus its citation). The "Attributes (F1)" window contains text boxes for editing text attributes of the currently selected element, while the "Attributes (F2)" window allows one to modify attributes that are constrained to selection from a defined list of values, such as a part of speech list (with the exception of single-selection lists, which also appear under "Attributes (F1)").

"Search (F3)" activates a search function which enables text searches to be made in the entire dictionary. "Format (F4)" allows certain settings to be modified that affect how the output will appear in the Preview Area and when exporting the dictionary data (for printing or electronic display). The filter function, "Filter (F5)", can be used to define criteria to work on or export a *subset* of the dictionary. The "Corpus (F6)" sub-window offers an integrated corpus query tool.

Preview Area [4]

This window displays an approximate representation of how the currently selected entry will appear in print (a so-called WYSIWYG ("what you see is what you get") view), as well as displaying the articles immediately following the currently selected article. The Preview Area updates immediately as changes are made in the Tree View or Attributes (F1 and F2) and Format (F4) sub-windows.

One very useful feature of the Preview Area is the displaying of *related cross-references* of the currently selected article – all article that are cross-referenced *by* the current article, as well as all those that have cross-references *to* the current article, are displayed in the Preview Area.

> **Printing:** Note that you can print the contents of the Preview Area (e.g. currently selected entry) by right-clicking in it and selecting "Print..." or "Print Preview...", or clicking the Printer icon in the Preview Area toolbar.

Bilingual "Side-by-Side" View Mode

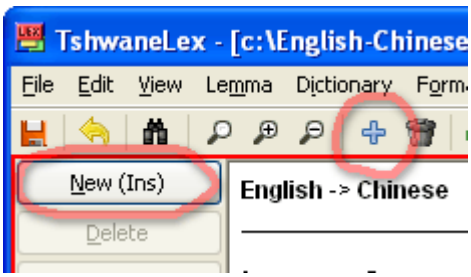
For a bilingual dictionary, all four main editing areas are shown for *each language*, side by side. Each of these sides (containing the four main areas) is known as a "language editing window", and the *currently active* language editing window is highlighted with a red border. (When one uses keyboard shortcuts such as "Esc" for the quicksearch box or "Insert" to add a new entry, these commands are activated for this currently active language editing window.)

"Expanded View" Mode vs. Side-by-Side View Mode

When working on a bilingual dictionary, the currently active language editing window may be expanded to occupy the entire TLex work area (as is always the case with the monolingual editing interface) by selecting the "Window/Expanded view" (Ctrl+W) menu option. This is useful for increasing the size of the work area when working mostly on one side of the dictionary.

To return to Side-by-Side View mode again, simply select this menu option (or press Ctrl+W) again.

Adding New Entries to the Dictionary/Termbase



The simplest way to create a new entry is to click on the "New (Ins)" button in the top left corner of the language editing window, or the "+" sign in the toolbar (see image at right). You will then be prompted for the headword/term of the entry to be added. Enter the headword and click "OK" or press the "Enter" key. A new, empty entry will be added, automatically correctly sorted. (If the new lemma is a homonym, the homonym number will also automatically be added by TLex.) To add word senses to the lemma, right-click on the "Lemma" element in the Tree View and click on "Add: Sense". To add a definition or a translation equivalent (TE) to the *sense*, right-click on the newly added "Sense" element in the Tree View and select "Add: Definition" or "Add: TE", and then edit the definition or translation equivalent under "Attributes (F1)". Alternatively, as a shortcut, one may left-click on the "Sense" element and then enter a definition or translation equivalent directly into the "°Definition (NEW)" or "°TE (NEW)" textboxes under "Attributes (F1)".

You can also use the "Lemma/Insert new lemma" menu option to add new articles to the dictionary, or use the "Insert" **shortcut key** on the keyboard. Note that when working on bilingual dictionaries in Side-by-Side View mode, these shortcuts will apply to the *currently active* language editing window, i.e. the side of the dictionary highlighted with a red border.

Structuring Articles with the Tree View Control

Adding: Right-clicking on any element in the Tree View shows a list of all possible elements that may be added as a child of that element. (Underneath the separator, cascades of child elements are also available. For example, instead of first adding the child element "Sense" to a "Lemma", followed by the addition of the child element "Example" to that "Sense", one may select "Add: Sense::Example" to obtain the same result with a single instruction. These "multi-element cascades" are shown with double-colons "::".)

Deleting: Elements can be deleted using the right-click "Delete" menu option, or by pressing "Delete" on the keyboard, to delete the currently selected element.

Moving: Right-click options to "Cut" (Ctrl+X) and "Paste" (Ctrl+V) elements are also available. A pair of scissors "[8<]" is shown in the Tree View during the process of cutting and pasting, and will remain visible in front of the last element pasted (to show what is currently on the clipboard).

The right-click "Cut" (Ctrl+X) and "Paste" (Ctrl+V) menu options can also be used to cut and paste elements between *different* articles.

Drag-and-Drop: It is also possible to move elements in the Tree View around by dragging them with the mouse cursor. This is disabled by default, but can be enabled as a user preference with the "View/Tree View drag-and-drop" menu option.

Copying: A right-click menu option "Copy" (Ctrl+C) allows an element subtree to be duplicated anywhere else in the document via "Paste" (Ctrl+V).

When a particular element is selected, it can be turned into a sub-element by choosing the right-click "Make sub-element" menu option. This will typically be used to change senses into subsenses.

A particular translation equivalent ("TE") can be turned into the first translation equivalent of a new sense with the right-click "Make separate sense" menu option.

Changing the Order of Elements

Elements on the same level in the tree hierarchy can be moved up and down, using either the right-click "Move up" and "Move down" options, or their corresponding **keyboard shortcuts** "Ctrl+Up" and "Ctrl+Down" respectively.

Changing the Order of Homonyms

To change the order of homonyms, right-click on the top-level "Lemma" element and use the right-click commands "Move homonym up" (Ctrl+Up) or "Move homonym down" (Ctrl+Down). The entries will be re-ordered and the homonym numbers will automatically update.

Marking Entries as 'Incomplete'

Entries that require further attention may be marked as "incomplete". The complete/incomplete status of each article can be toggled with the "Incomplete" checkbox under "Attributes (F1)", or by pressing Ctrl+Shift+I. Incomplete entries are displayed with question marks next to them in the Preview Area.

Various import and export functions of TLex/tlTerm make use of the "incomplete" setting. For example, when exporting data in preparation to be printed, incomplete entries are by default (this is a configurable option) excluded from the output, to help prevent you from inadvertently publishing unfinished entries.

Note that the "Filter (F5)" tool can easily be used to find and show all entries marked as incomplete at any time.

Saving Your Work

Newly created or modified entries are displayed with a "*" (or "!") next to them in the Preview Area. This means "unsaved changes", and appears on all new entries which have changes that have not yet been saved to disk. To save your changes, select "File/Save" (Ctrl+S). It is a good idea to do this regularly.

Restoring the Last-saved Version of an Entry ('Undo')

If, when working on an entry, you change your mind about the changes, it is possible to "restore" that entry to the *last saved version*. To do so, choose the "Edit/Restore" menu option (Ctrl+Shift+Z). This command will only have effect when there are unsaved changes in your entry, i.e. when a "*" appears next to that entry, and when the entry has been saved to disk at least once before.

Adding a Batch of New Entries, Or 'Import Wordlist/CSV'

It is possible to add many entries to the database at once from a wordlist (one word per line), or from a CSV (Comma Separated Values) file from spreadsheet software such as OpenOffice.org or Microsoft Excel. To do this, use the "File/Import/Wordlist or CSV (Comma Separated Values)" menu option. See the chapter "Importing Data into TLex" for more information.

Deleting Entries from the Dictionary/Termbase

To remove an entry from the dictionary/termbase, select that entry in the Entry List, and click the "Delete" button (near the top left of the editing window). You can also use the keyboard shortcut "Ctrl+Delete".

Note that in TLex, if you have other entries that contain "smart cross-references" to the entry you are deleting, TLex will warn you about these before deleting the entry. TLex will also automatically remove these cross-references for you if you proceed to delete the entry.

Printing your Dictionary

When you wish to print your dictionary, the simplest is to export it to RTF (Rich Text Format) and open it in any word processor, such as OpenOffice.org, Corel WordPerfect or Microsoft Word. Apply any final formatting changes (e.g. adding columns) in the word processor before printing. To export to RTF format, select the "File/Export/RTF (Rich Text Format)" option from the menu. An "Export options" dialog will appear (do not worry about these options at this stage, their meaning will be explained in the course of this User Guide). Click "OK". You will be prompted to enter a filename for the output RTF file. If you are working on a (semi-)bilingual dictionary, you will be prompted for a filename twice – once for each side of the dictionary.

Note that by default, articles that have been marked as "Incomplete" (i.e. those that are displayed with question marks) will not be exported. This can be overridden, if desired, in the "Export options" dialog box, with a further option to hide the incomplete marker "?".

Making Regular Backups

Manual Backups

For all dictionary and terminology projects, it is **crucial** to make regular backups. TLex/tlTerm provides a feature that makes it easy to quickly create a backup of your current database. To create a backup copy of your database at any time while working on it, select the "File/Create a backup"

menu option. This will save a full "snapshot" of your database into a backup folder. (Note that, by default, this will typically be saved on the same computer as the file you are working on. This will thus not protect you if that computer is damaged or stolen; it is crucial to also make "offsite" backups, discussed below.)

The folder in which backups are stored may be configured using the "settings" dialog, which may be opened using the "Tools/Options" menu option.

Automatic Backups

In addition to the manual "File/Create a backup" menu option, TLex/tlTerm also has an *automatic* backup system that automatically saves a backup copy of the currently open database into the backups folder at a configurable time interval. By default, this is set to one hour ("60" minutes). Whether or not to perform automatic backups, and the interval in minutes, can be configured under the "Tools/Options" menu (on the "settings" tab labelled "General"). Every 'interval' the previous backup is overwritten with the latest backup, but one backup each per twenty-four hours is kept.

All users are also very strongly encouraged to periodically make *offsite backups*, discussed next.

Offsite Backups

The backups created manually using "File/Create a backup" as well as the "automatic backups" will by default be saved **to the same computer** on which you are working (although you may configure this to save to a "mapped network drive" on another computer on the LAN (Local Area Network)). However, to protect against disastrous eventualities such as hard disk failures, theft, lightning, earthquakes, fires, nuclear incidents and so on, **it is crucial** to also have a policy for creating regular *offsite backups* – that is, backups that are stored at least in a *different building* to the one in which the computer being used to compile the dictionary is. This could be as simple as regularly writing a copy of your database file to a CD or flash disk and taking it to someone's home, or e-mailing a copy of the database to a colleague. You may wish to consider further protecting offsite backups by storing them in a safe.

It is recommended that you make an offsite backup at least once every two weeks.

"Cloud" Backups

Many service providers (such as 'DropBox') now provide automatic file storage services onto their servers on the Internet, with easy-to-use software that automatically uploads files from selected folders on your computer. This can provide an additional convenient offsite backup system for you. However, **do be careful** – if the only copy of your document is in that folder for "cloud" synchronisation, you can still lose the data if a corrupted copy is uploaded, as most such services do not maintain full file histories. Thus, such services may lull you into a false sense of security with respect to your data – use with caution as an *additional* backup facility. Also, services like DropBox are not implemented with 'true' encryption-based data protection – for an alternative that is, consider 'SpiderOak'.

NB: An additional "security caveat" also applies for projects in which confidentiality is critical. Not all such services are as secure as one might expect!

Getting Started with tlTerm: QuickStart Guide

This 11-minute “Getting Started with tlTerm” video is also a useful introduction:
https://youtu.be/KBHRFtkCOag

Creating a New Termbase

To create a new termbase, open tlTerm, and select “File/New terminology list”, or click on “Create a new terminology list” in the main 'Tasks' window on the Start Page. The 'create termbase' dialog appears:

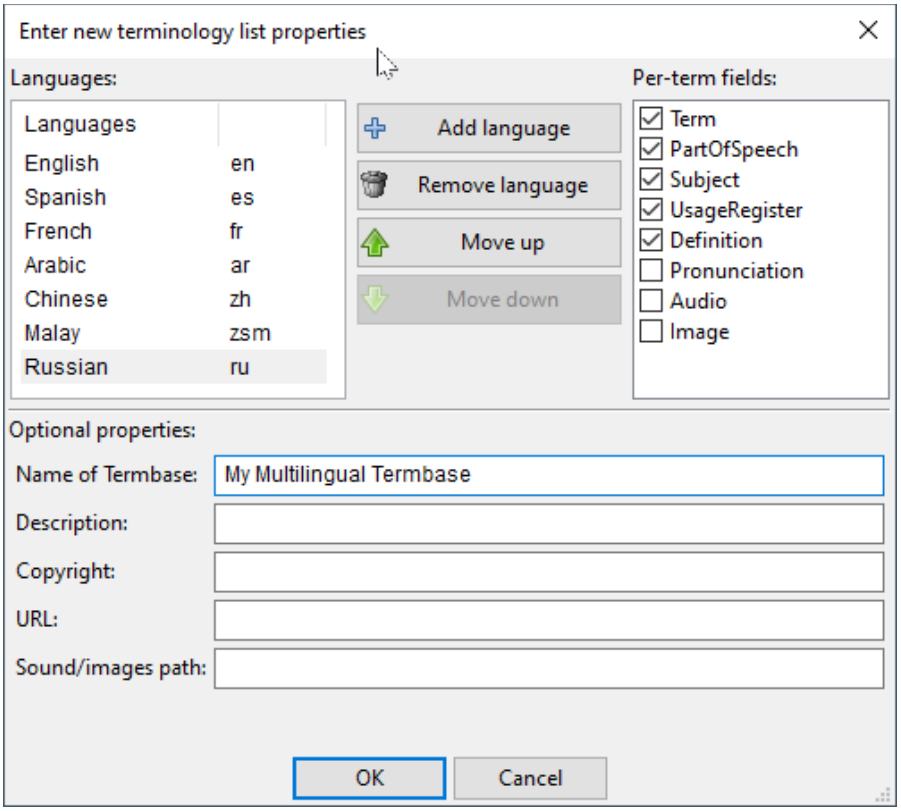


*Figure 4: The 'create new termbase' dialog in tlTerm. You may specify as many (or as few) languages as you like. More languages may be added later via the **Termbase/Properties** menu command which brings you back to this dialog.*

The “**Add language**” and “**Remove language**” buttons can be used to enter the list of languages that you would like to treat in your termbase. The field list on the right can be used to select which *types of information* you would like to be able to enter *for each term in each language*. The various other fields are optional.

Most of the basic editing functionality in tlTerm is similar to TLex; please proceed by working through the “Getting Started with TLex” section. Terms are added in the Tree View.

Tip: You can return to this dialog later again via the “Termbase/Properties” menu command

Once you click “OK”, tlTerm then asks if you want to “Enable user management” (if desired). This is optional; you may skip this step. (You can use the “Termbase/User management” menu command to return to these settings later.)

After that, you can start working on your terminology list.

It is suggested to now do 'File / Save' (or Ctrl+S), which will prompt you for a local filename (it uses the file extension ".tterm") to save your terminology database to. You can work on it again later by just opening this file. (If you want to create a multi-user setup, see the section on ODBC.)

Important notes: In tlTerm and TLex, you *can* enable "User management" (i.e. having to log in with user login names and passwords) *even when working on local files*, and this can be very useful: It is suggested to do so if you want the software to track **created-by**, **last-modified-by**, **last-modified-date** and **created-date** information on entries.

NB: This can be enabled or disabled later at any time via "Termbase/User management" (in tlTerm) or "Dictionary/User management" in TLex (though it is compulsory for ODBC setups).

If you enable user management, the option "Allow database to be opened without a password" allows a 'guest user' to open the file for reading by pressing "Escape" at the login prompt.

Lost passwords: Do not forget your password(s). If you forget your password, we may be able to recover lost passwords, but we may charge a small fee to do so (typically 1 or 2 hours of support at standard rates). (Note that in future versions of the software, it may become impossible to recover lost passwords.)

Primary Differences Between tlTerm and TLex

Terminology is Concept-Centric

The main difference between tlTerm and TLex is that terminology is *concept-oriented*, while lexicography is *lemma-sign ('word' / phrase) oriented*. (This slightly affects how you edit entries.)

In terminology, an entry in the database generally corresponds to an *abstract concept*, e.g. you might have a database entry for the 'abstract concept' of a "cat" in general, i.e. not associated with any particular language. The terms for that concept in each of the various treated language are then attached, as "equals" and on the same level, to that central concept. Multiple terms that are synonyms in the same language would be attached to that same entry/concept, as they are merely additional terms for the *same concept*.

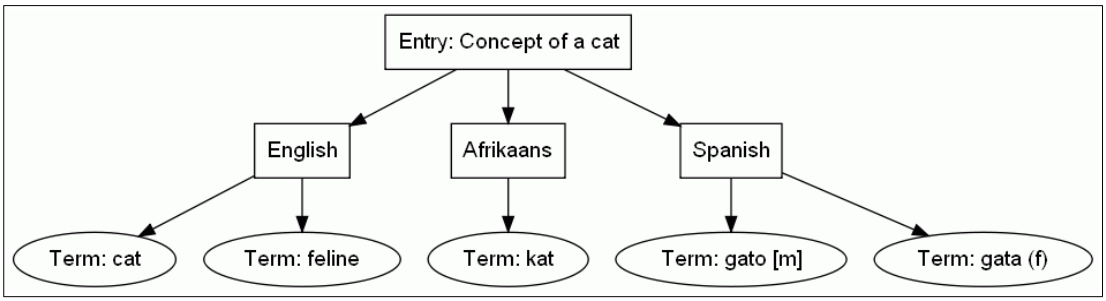


Figure 5: Conceptual structure of an entry in a terminology database

In lexicography (and thus TLex), an entry in the database corresponds to an actual *orthographic word* in a *particular language*, e.g. the actual "English word **cat**". Any translation equivalents in

other languages may then be attached to that word – like a "hub and spoke" model. Synonyms of that word in the same language (e.g. English "feline") would be treated as *entirely separate entries*, (unlike tlTerm, where they would belong on the same entry for the 'cat concept'), perhaps with cross-references to associate the two (e.g. a reference "see **cat**" under the entry for "feline").

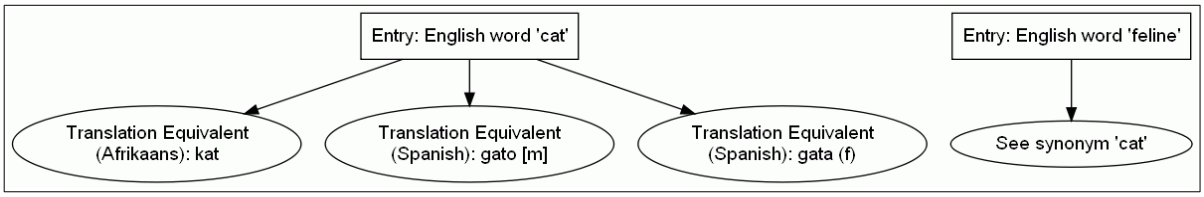


Figure 6: Conceptual structure of two synonymous entries in a dictionary database

tlTerm, unlike TLex, also contains an **extra drop-down list** in the main toolbar, from which you can select which language you want to view / index (sort) the main Entry List on the left by. Entries (concepts) that do not yet contain any terms in the chosen language for that entry/concept are displayed in the list as a "-". **NB: This makes it easy to find untreated entries** for a particular language. (This is similar to the "Sort by" function under "Format (F4)".)

Adding Synonym Cross-References in tlTerm vs TLex

Since an entry in a *terminology* database corresponds to a *concept*, it follows from this that a synonym of a word/term/phrase belongs to the *same concept*, and therefore in tlTerm if we want to add synonym "cross-references", we just add additional translations/terms *to the same entry*, as second or third or fourth translations/terms in the same language (e.g. "dog" and "hound" might both be added as English terms under the entry for the concept of "dog").

Thus in tlTerm, we don't directly structurally add actual cross-references *per se*. tlTerm 'knows' these are synonyms within the same entry (and thus same concept), and automatically displays the extra terms in your Entry List as you work (e.g. both "dog" and "hound" appear in your Entry List, but if you click on them, they both take you to the *same actual entry* for the single concept). When exporting your data (e.g. to Word/RTF/HTML), it will also *generate* a special ***automatically-generated entry*** alphabetically under "hound" that has a reference to "dog". So it may look like there are two *entries*, but there is just one entry for the concept "dog".

In TLex, adding cross-references is done very differently: Since entries in a *dictionary* correspond to orthographic words/terms/lemma-signs, you will create two entirely separate entries - one for "dog", one for "hound" - and if you want to create a synonym cross-reference from "hound" to "dog", this would typically be done by adding a "References" smart-references element in the Tree View that 'points' from one entry to the other. (The precise details, e.g. names of elements, may vary slightly depending on your project, but the general approach would be the same.)

These are the main differences between TLex and tlTerm. Other than these, day-to-day editing is largely done in a very similar way, and most the features of the software correspond very closely.

Adding Custom Fields

Both TLex and tlTerm fully allow "Customising the DTD" (see the relevant chapter on this for details), i.e. you can add your own fields (that is, define your own custom elements and attributes).

(However, keep in mind that if doing *terminology*, if you add custom fields it may be a bit more difficult to load that information in other terminology software if you need to, as the other

terminology software might not understand or be able to handle those custom fields. If you are not expecting to interchange your data with other terminology software, this is not a problem.)

Snapshots and "Show Differences" (Review Changes)

Introduction to Snapshots

When you start editing an entry, the snapshots system creates a temporary local 'snapshot' of how the entry looks the moment you start editing.

You can then review your local unsaved changes while editing by clicking "Show Diffferences" (in the 'snapshots toolbar').

This is also extremely useful for reviewing changes before saving, to help make sure there are no unintended changes.

Functionality related to snapshots is represented by the "📷" symbol.

> Snapshots are local only to your computer, and are temporary. By default they are not saved anywhere.
>
> It is possible to save/load snapshots if needed, though that would generally only be for more advanced purposes.

> **Note:** Snapshots are presently based on how the *output* of the entry looks. More functionality is being added for also working with snapshots based on e.g. the underlying internal XML data structure and for potentially in future working with snapshot histories.

More Advanced Snapshots Functionality

Please note for some of the more advanced snapshots functionality, it may be helpful to understand the basics of filters (see "Find/Filters (F5) section).

Filtering to Show Entries With Differences (Review Differences to Multiple Entries)

Clicking "All with diffs" will set up and apply a special filter that shows only entries that have *differences in their **output*** compared to their *local snapshots*.

You may navigate these with the usual filter commands to 'go to next' etc. to work through these, or use commands like "Next+Diffs" in the snapshots toolbar to jump to each next entry with diffferences, showing the differences, allowing you to quickly review local changes to *multiple* unsaved entries.

This is also very useful before saving to help review changes if you have multiple entries with unsaved changes.

[Advanced] Creating and Saving Sets of Snapshots

If you right-click on the entry list, under the menu commands with labels starting with “Apply action to ... entries”, there are commands labelled “Create instant full snapshot backup”. These allow you to save text, HTML or XML ‘snapshots’ for either all entries, or subsets of entries.

This can be very useful for e.g. programmers/administrators for helping to compare and review changes to the entire database before and after, say, running a Lua script that does some data remodeling, as it can be used to help check and verify that the output of the entries remains the same (and thus that your script didn’t ‘break anything’). In this case the comparison would be done using tools outside of the software (e.g. Total Commander’s text/folder comparison).

An editor may also use this to help save sets of ‘snapshots in time’ to help compare and review entries against past ‘historical’ versions of your entries.

These snapshots in this case are saved as external files and not ‘attached’ the database in anyway.

Changing the Interface Language (Localisation)

The interface of TLex itself can be configured via “Tools/Options”, under “Language”.

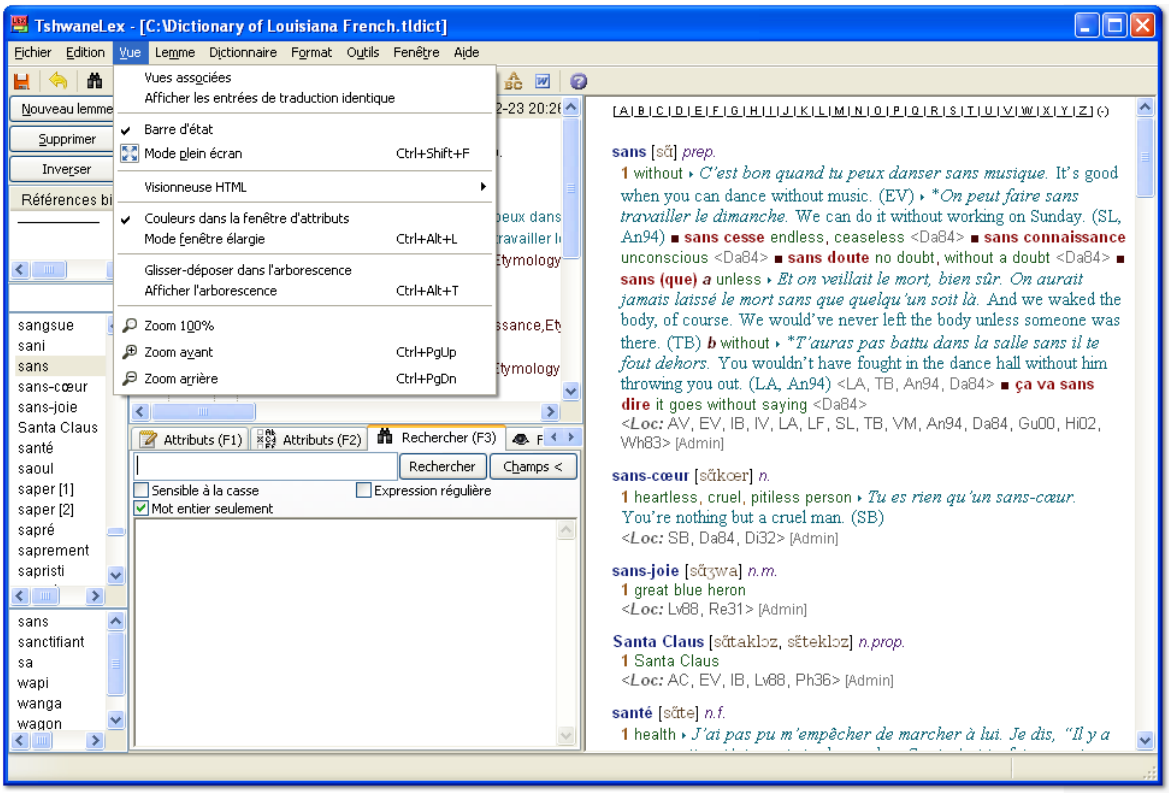


Figure 7: TLex with French Interface (data Copyright Dictionary of Louisiana French Project)

Built-in Localisation Editor – Creating New Interface Languages

TLex/tlTerm include the tools to change and edit their own interface language. These can also be accessed via “Tools/Options” under “Language”. This tool is fairly straightforward, and can be used to create new interface languages from within the software itself, by clicking on the “New” button, then using “Save as” when the localisation editor appears. The localisation editor will display an interface showing the original English strings, and a box where you can enter the translated string. Each string is also rooted to a “key” value, which looks something like “MENU_FILE” - this gives a clue as to where it is used in the software, e.g. all “MENU_” strings appear in the application main menu.

The 'Apply' button can be used at any time while translating to apply the new translations immediately (for some of them, however, a restart of TLex or tlTerm is still required).

When you click “Save”, the translated strings are saved in a “.lang” type file under a folder that will be something like “c:\Program Files\TLexSuite\Data\Catalogs”. This file can be distributed to other machines by just copying it into that folder. (It is in fact a simple text file.)

There are a few “special” characters and strings relating to the localisation:

& Appears in front of the character that will typically become a shortcut key to access the command, e.g. “&File” to make “Alt+F” open the “File” menu (in Windows, these may be underlined).

\t Indicates a “tab” character. This must typically appear in menu items between the command name and the shortcut key, e.g. “&Save\tCtrl+S” for MENU_FILE_SAVE. You enter a tab character by pressing “Ctrl+Tab”.

\n Newline.

%s Indicates that the software will, at run-time, substitute this with a string value, e.g. “Enter new text for label "%s"”.

%d Indicates that the software will, at run-time, substitute this with a numerical value.

Note that the relative ordering of %s and %d markers *within one string* must remain the same.

Keeping Your Software Up To Date

TshwaneDJe continually releases new “maintenance updates” for the software. These updates contain important and useful improvements and bugfixes, and it is strongly recommended that you keep your software up to date. You can check if there is a newer release available for your version of the software by using the “Help/Check for updates” menu option. This will open your web browser to a site that tells you if a newer version is available, and if so, tells you where and how to download it.

Maintenance updates (not to be confused with actual version *upgrades*) are free. When you install a maintenance update, re-activation is not usually required.

“Wide View” Mode and “High View” Mode

During compilation, it may be desirable to change the work area of the Attributes and Tools sub-windows (F1 to F6).

Encyclopaedia entries, for example, are typically much longer than dictionary articles, and one may want more “horizontal” space for the various input boxes. With the “View/Wide Tools window layout” menu option (Ctrl+Alt+L), the Attributes and Tools sub-windows (F1 to F6) can be widened to cover most of the width of the TLex/tlTerm window. This is illustrated for the compilation of encyclopaedia entries in the screenshot below:

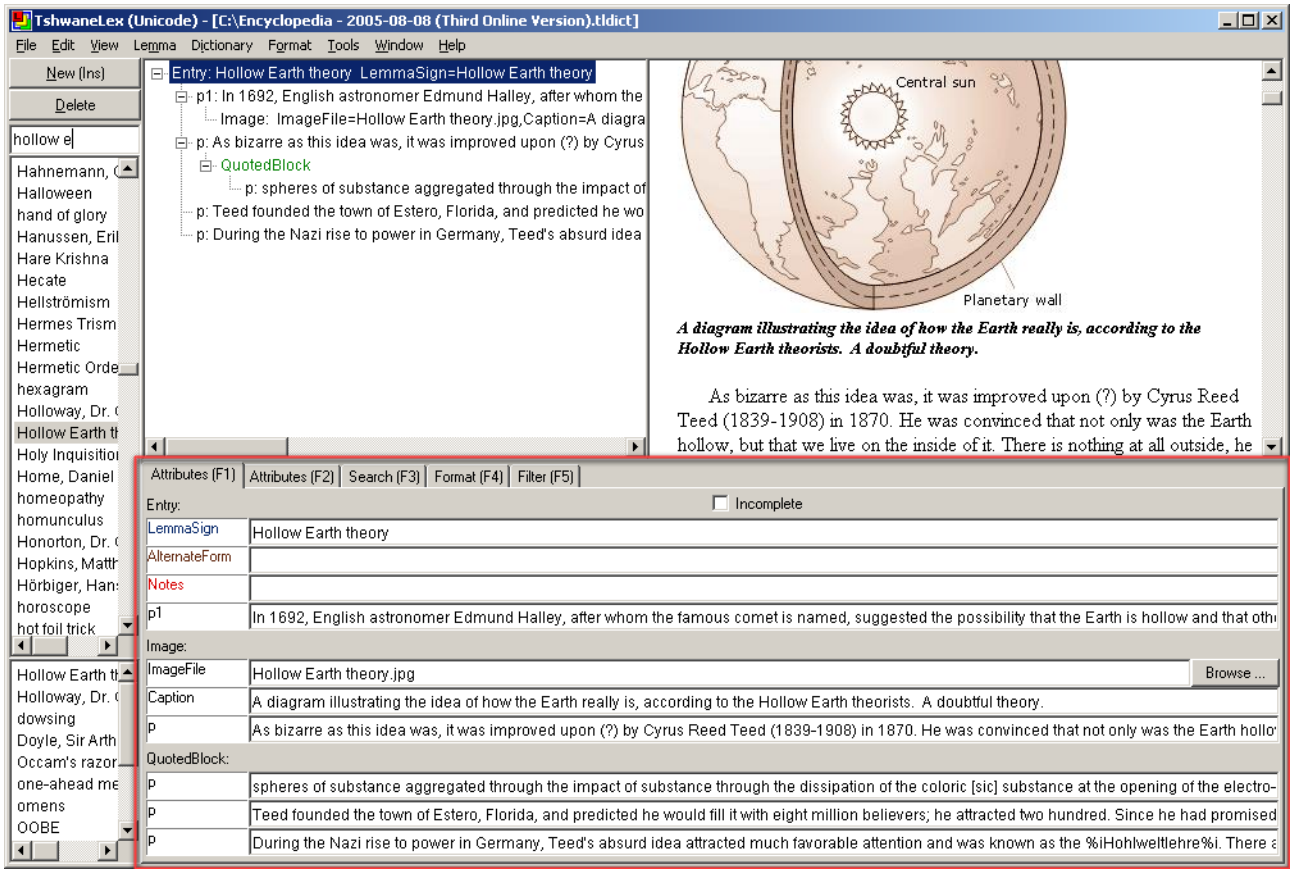


Figure 8: *“Attributes (F1)” in wide view window layout (Ctrl+Alt+L) for James Randi’s Encyclopedia [Data online at: http://randi.org/encyclopedia]*

In other cases one may wish to see more “vertical” space, so as to for instance work with more input boxes under “Attributes (F1)” at a single glance.

This is typically the case in multilingual terminography. With the “View/Toggle Tree View” menu option (Ctrl+Alt+T), the Tree View can (momentarily) be hidden. In the screenshot below, for example, the high view window layout has been enabled for the compilation of an eleven-lingual AIDS terminology list.

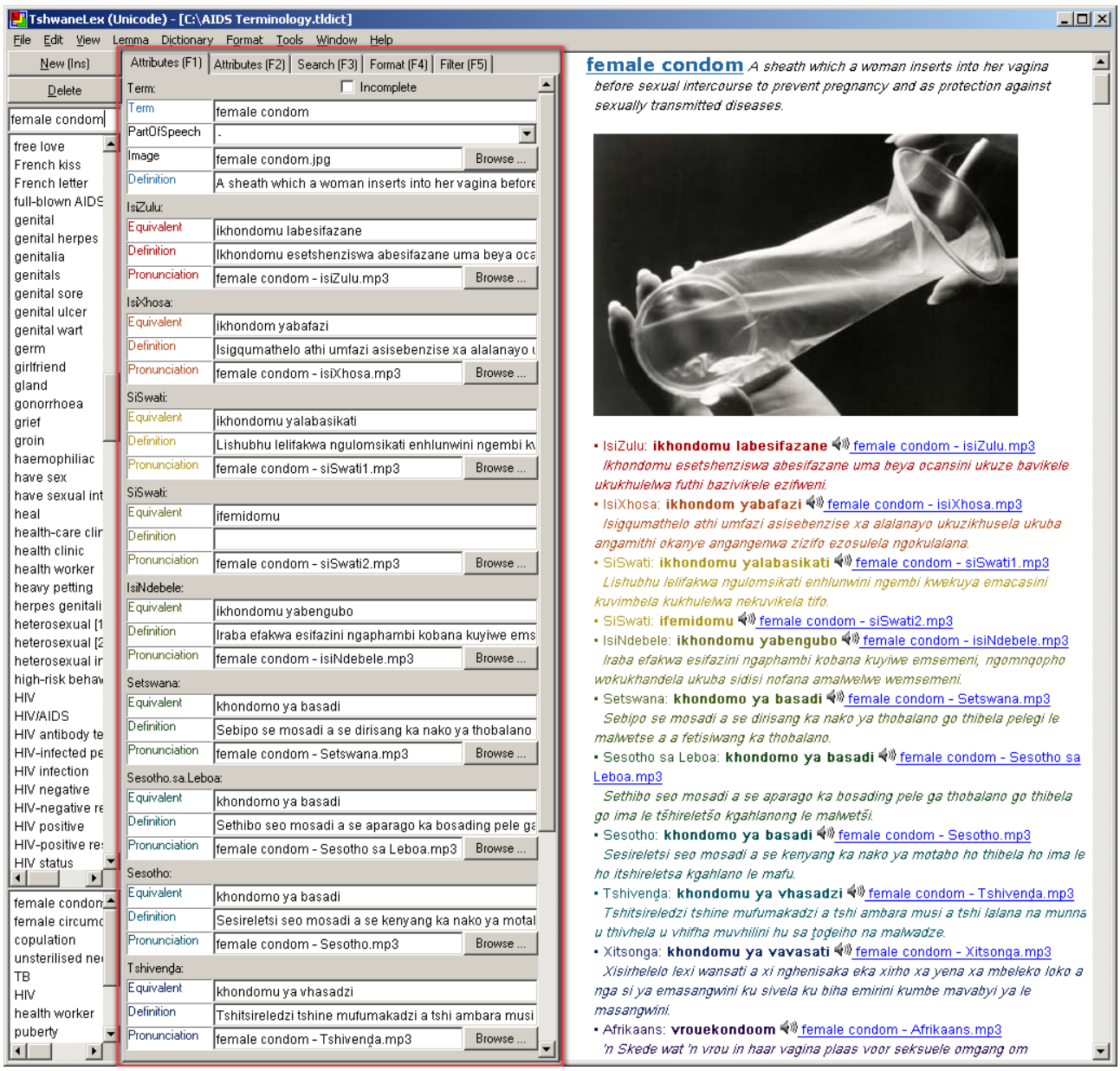


Figure 9: *"Attributes (F1)" in high view window layout (Ctrl+Alt+T) for an eleven-lingual AIDS terminology list [Data: © Department of Arts & Culture, South Africa]*

Note that although the wide and high view window layouts were illustrated for the "Attributes (F1)" sub-window, these views are also available for all other sub-windows (F2 to F6).

F12 Overlay Window (Larger Attribute Editing Window)

In addition to the "wide" tools window layout, another option to gain more editing space for editing the content of text boxes is the **F12 overlay window**. While in an "Attributes (F1)" text box, pressing F12 will pop up a larger window for editing the given attribute. Once done, pressing F12 closes the window again. While the F12 window is open, you can select any "Attributes (F1)" window, and the F12 window will automatically switch to editing that attribute. This window also allows *newline characters* to be entered in a more intuitive way.

The F12 window also works in various text edit boxes elsewhere in the software. (This can be useful particularly when editing certain types of text values, such as a Lua script.)

Attributes (F1)

The majority of a dictionary's article contents are **typed** into the various boxes under "Attributes (F1)". This is the primary purpose of the F1 sub-window; however, one also edits *drop-down closed list* attribute values here. Drop-down lists allow *only one* possible value to be selected at a time. In the screenshot below, TLex is used as an address book, and the value of the "Country" attribute is being selected from a drop-down list.

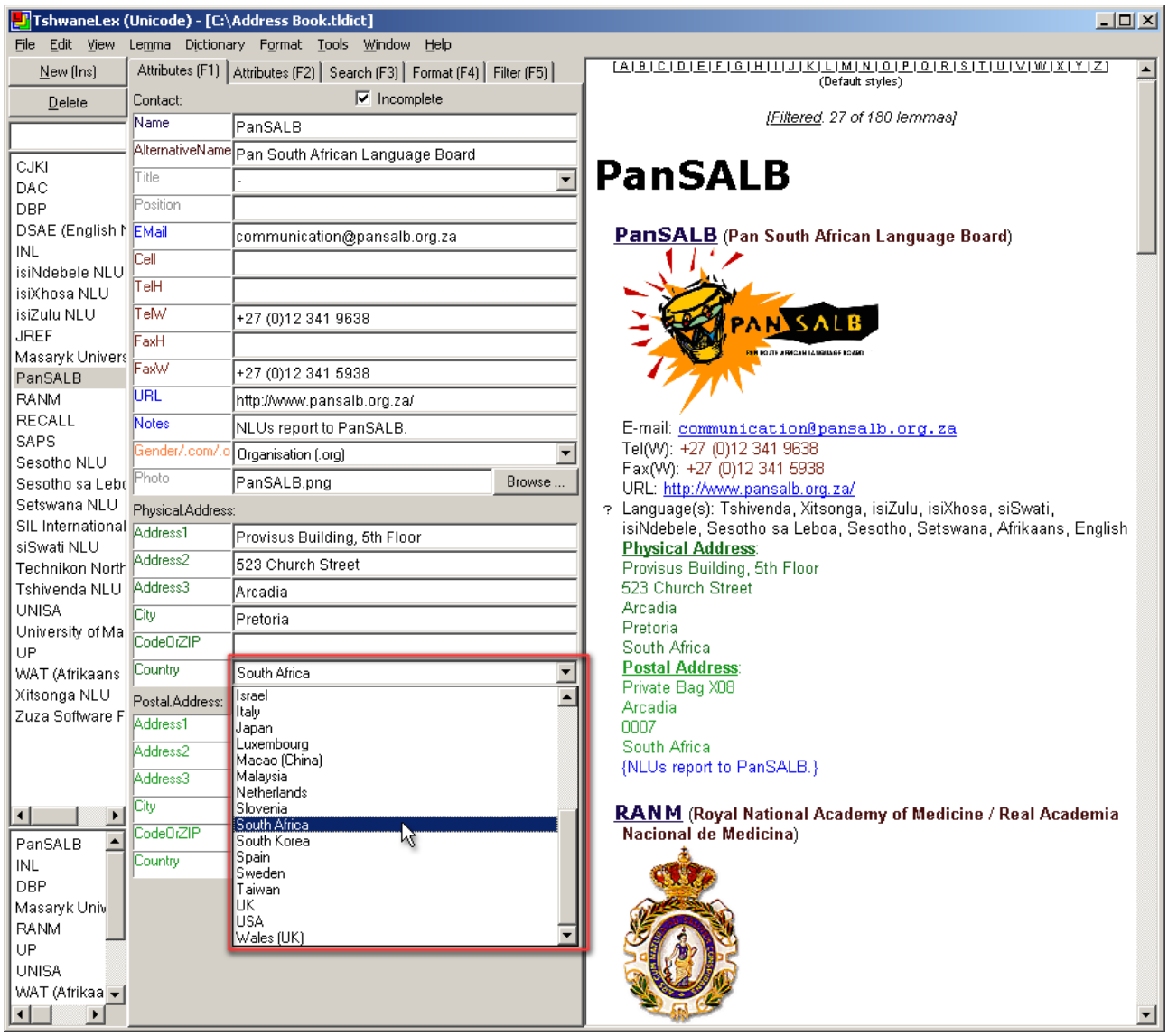


Figure 10: *Selecting an attribute value from a drop-down list under "Attributes (F1)", in an address book*

The way in which such attribute lists are set up is discussed in the section on Attribute Lists, in the chapter on Customising the Dictionary Grammar using the DTD.

By default, the attribute labels under F1 are displayed in the same colour as the colours for those attributes in the article Preview. This can be disabled with the "View/Use colours in Attributes window" menu option.

"Incomplete" Checkbox

Note that in the previous screenshot, the Contact "PanSALB" has been marked "Incomplete", hence the question mark which appears on the left of the entry in the Article Preview. See the section on 'Marking Entries as Incomplete' for more information.

Attributes (F2) – Closed Lists with Multiple Selection

The "Attributes (F2)" sub-window displays attributes whose values can be entered by simply **ticking** one or more values from a defined list. In the screenshot below, for example, two parts of speech have been attached to the lemma sign.

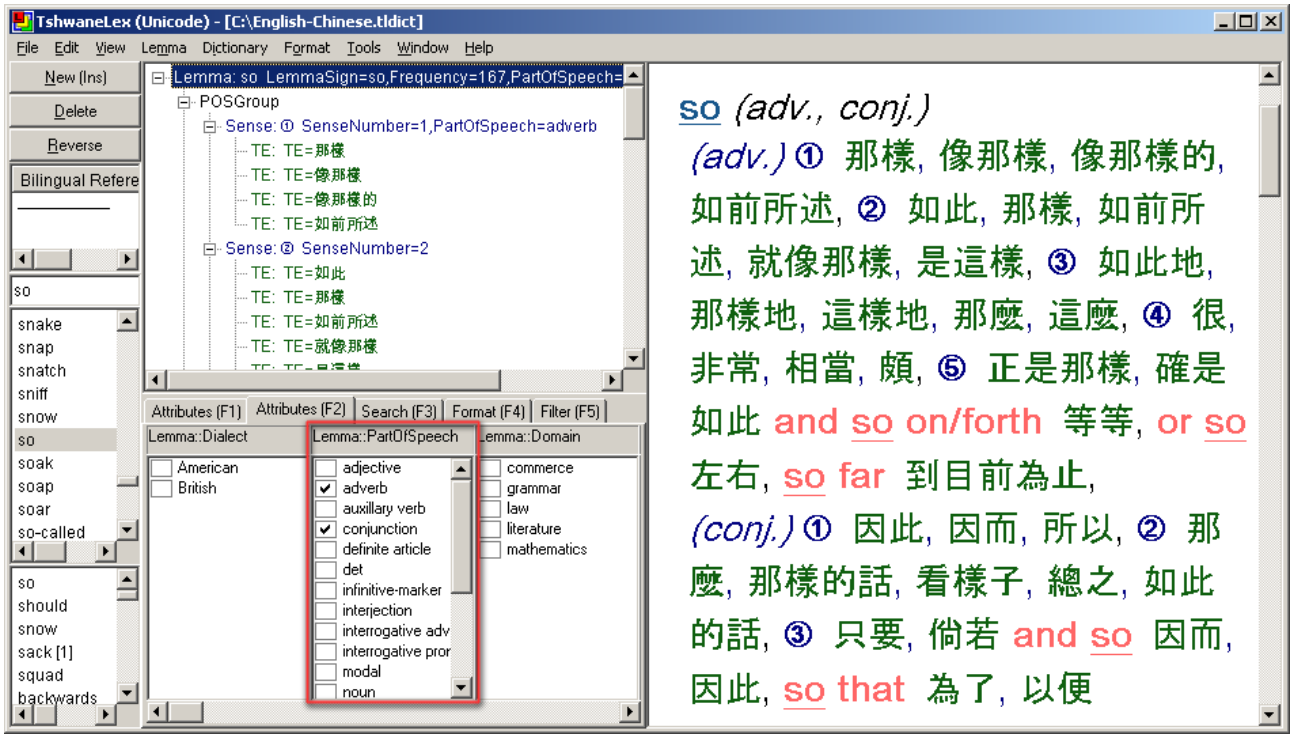


Figure 11: *Selecting multiple attributes from a closed list under "Attributes (F2)" in a bilingual English – Chinese dictionary [Data: © Lorraine Liang]*

Under F2, two types of multi-selection attribute lists may appear: "multiple, unsorted" and "multiple, sorted". "Unsorted" attribute list values will be displayed in the output in the same order as the user ticked those values; "sorted" will always be displayed in the output in the same order the list items appear in the list. The way in which these attribute lists need to be set up is discussed in the section on 'Attribute Lists' in the chapter on Customising the Dictionary Grammar using the DTD.

Expansion of Tilde Character (~)

The tilde character "~" can be used within any attribute as a substitute for the lemma sign (headword). This is typically used to represent the lemma sign in usage examples, as well as in idioms and other multi-word units, in order to save space in print dictionaries, as shown in the example below:

draw *(verb)* to create a picture or diagram, usu on paper : ***to ~ a picture*** ► **~ the line** define or establish a boundary or limit : ***"We ~ the line at swearing"***

Although this saves space, it is less user-friendly for the user of the dictionary; thus when publishing to electronic or online media, the tilde can automatically be expanded by TLex and be replaced with the lemma sign of the article, e.g.:

draw *(verb)* to create a picture or diagram, usu on paper : ***to draw a picture*** ► **draw the line** define or establish a boundary or limit : ***"We draw the line at swearing"***

This behaviour can be turned on or off using the "Expand tildes (~)" checkbox option under "Format (F4)". Changes are immediately visible in the Preview Area.

Overriding the Tilde Substitution Text

In some situations, the form of a word in a usage example or multi-word unit differs from the form of the lemma sign itself. This results in the incorrect form being automatically substituted, e.g.:

university *(noun)* an institution of higher learning with teaching and research facilities that is authorised to grant academic degrees : ***She graduated from the university of London in 1981.***

TLex provides a solution to this whereby you can *override* the text that the tilde is replaced with in specific cases. This is done by entering the desired text to be substituted in square brackets directly after the tilde, e.g.:

```
She graduated from the ~[University] of London in 1981.
```

The output will then appear correctly, as follows:

university *(noun)* an institution of higher learning with teaching and research facilities that is authorised to grant academic degrees : ***She graduated from the University of London in 1981.***

Using Text Formatting Within a Text Field

Note that general formatting issues are discussed in the chapter on the Styles System; this section deals with overriding formatting *within* attributes.

Markup Characters (Bold, Italics, Underline etc.)

TLex (and tlTerm) provides special **formatting markup characters** that may be used to specify formatting styles such as bold, italics and underline within a *subsection* of an attribute. These are specified by entering a "%" symbol followed by the markup character for the desired formatting style, e.g. "b" for "bold". For example, if one wants the following to appear in the output:

> This example demonstrates the use of **bold** and *italics* within an attribute.

Then one would enter the following:

> This example demonstrates the use of %bbold%b and %iitalics%i within an attribute.

Following is a full list of the markup characters available in the software, as well as corresponding shortcut keys which may also be used to enter them:

Markup Character	Meaning	Shortcut Key
%b	**B**old	Ctrl+B
%B	**B**old	–
%i	**I**talics	Ctrl+I
%I	**I**talics	–
%u	**U**nderline	Ctrl+U
%r	Superscript (**r**aise)	–
%l	Subscript (**l**ower)	–
%k	Stri**k**ethrough (strikeout)	–
%s	**S**mall caps	–
%n	**N**ew line	–

In order to generate an actual "%" symbol in the output, use "%%".

All of these markup characters, except for "%n" (New line), must appear in *pairs* that enclose the text being formatted. The software will display a large red error message if a markup character has not been "closed". Instead of physically typing the %b, %i and %u markup characters on either side, one can also select (highlight) the section one wants to markup first, followed by pressing %b, %i or %u respectively.

[TLex-specific] Markup characters can also be used in the lemma-sign field. For example, if one wishes a Latin lemma sign to be displayed in italics, say *sub judice*, then it may be entered as "%isub judice%i". TLex will still sort the lemma correctly.

Note that markup characters can also be used within the Styles System. See the chapter on the Styles System for more information.

Smart Formatting Reversal (%b, %i vs. %B, %I - "soft" vs. "hard" bold/italics)

For the markup characters "%i" and "%b", if the text surrounding the marked-up text is *already* italics or bold, respectively, then the formatting will revert to non-italics (Roman) or non-bold. For example, if one has the following text in an attribute:

As in Orwell's %iAnimal Farm%i, however, some people are "more equal than others"

Then this would ordinarily be output as follows:

As in Orwell's *Animal Farm*, however, some people are "more equal than others"

However, if the entire attribute is being output in italics, due to the Styles, then it would be output as follows:

As in Orwell's Animal Farm*, however, some people are "more equal than others"*

If you wish to "force" the marked-up text to *always* be italics or bold, regardless of the style of the surrounding text, then use the uppercase versions of the markup characters, "%I" or "%B", as in the example below:

%IHomo sapiens%I is a separate species from Neanderthals and other hominids

Using Inline Elements for Text Formatting [Advanced]

An alternative to the special percentage-sign markup for formatting, that is more closely based on XML, is to use "inline elements" (PCDATA, or "parsed character data"). Formatting for bold and italics would then take the form of XML tags; thus for example, instead of "%i", you might use "" and "" opening and closing tags for italics:

As in Orwell's Animal Farm, however, some people are "more equal than others"

Inline elements have certain advantages and are in many ways more powerful (and are more standard), but also have some drawbacks/limitations. They are explained in more detail in the online FAQ (Frequently Asked Questions: http://tshwanedje.com/faq.html).

Configurable Keyboard Shortcuts

In TLex/tlTerm, keyboard shortcuts, also known as macros, can be used to type in characters (with or without diacritics), or symbols, that are not available on your keyboard. By extension, such shortcuts can also be used to type in entire text strings. To create those shortcuts, go to the "Tools/Options/Keyboard shortcuts (macros)" menu option.

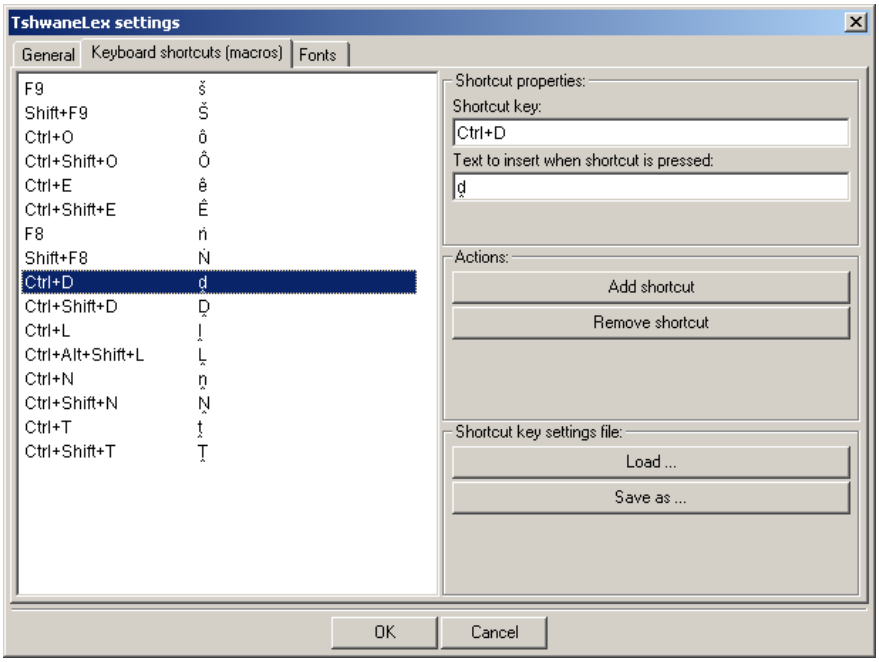


Figure 12: *TLex settings, Keyboard shortcuts (macros) tab*

From the above screenshot, one for example sees that when pressing the shortcut key “Ctrl+D”, the “Latin small letter D with circumflex below” character will be output.

It is suggested that such “special characters” are first selected in a word processor, using the “Insert/Special Character...” (OpenOffice.org) or “Insert/Symbol...” (Microsoft Word) menu option. It is important to choose a Unicode font in the word processor, in order for the special character to be “future proof”. One can then copy (Ctrl+C) the character from the word processor, and paste (Ctrl+V) it into the “Text to insert when shortcut is pressed” field of the dialog shown above.

If one knows the hexadecimal value of the character, one can also use the built-in “Tools/Character Information Tool” to “generate” the character, as illustrated below:

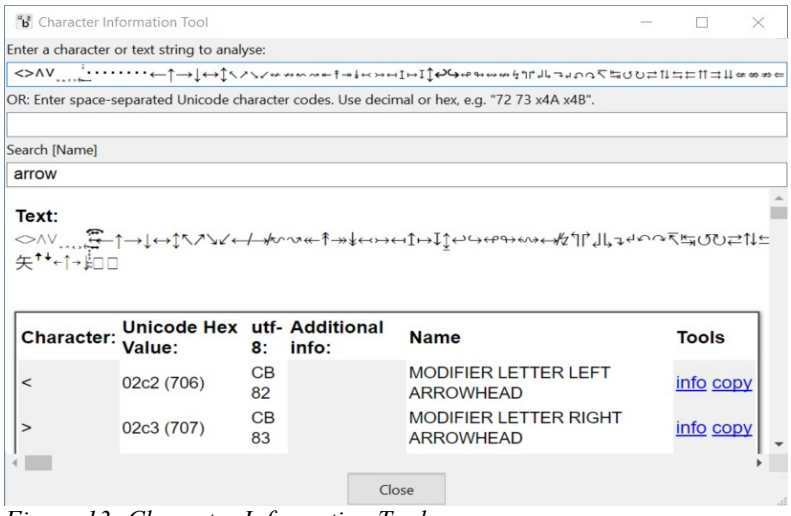


Figure 13: Character Information Tool

Note that, while in the “Shortcut key” input box of the keyboard shortcut configuration dialog, you have to *press the shortcut key* itself. Also, when choosing the shortcut key, you must use a shortcut that is not already in use for one of the menu commands (the shortcut key will not work).

New keyboard shortcuts can be added (“Add shortcut”), while existing ones can be removed (“Remove shortcut”).

It is also possible to save a set of keyboard shortcuts (“Save as ...”), and to load an existing set (“Load ...”).

Inline Elements: Special 'Tagging' Shortcut Keys [Advanced]

Special "tag" shortcut keys can be created under "Tools/Options/Keyboard shortcuts (macros)" that make tagging data with inline elements under "Attributes (F1)" far more convenient and user-friendly than typing out tags manually. This involves creating a shortcut key with the following format for "Text to insert when shortcut is pressed":

TAG:tagname

For example:

TAG:b

Pressing this shortcut key in an "Attributes (F1)" box will then automatically 'intelligently' output either an opening "" or closing "" tag as appropriate, or if some text is selected, surround the selected text with a pair of opening and closing tags.

Replace As You Type

In addition to the keyboard shortcuts, some “special characters” can also be typed in as pre-prepared “replace-as-you-type” combinations. For example, if you enter “(c)” it will automatically be replaced by the actual copyright symbol, “©”. If you enter “\+-”, it will be replaced by the symbol “±”. Various other combinations exist, including combinations for entering phonetic symbols. To see a complete list of the available “replace-as-you-type” combinations, access the “Tools/Show replace-as-you-type help” menu command. (This can be printed by right-clicking in the help window and selecting “Print...”.)

By default, “replace-as-you-type” is enabled. This setting can be toggled with the “Tools/Enable replace-as-you-type” menu command.

Configuring Replace-As-You-Type [Advanced]

The default “DefaultReplaceAsYouType.rayt” configuration file can be found in the application data folder (typically “C:\Program Files\TLexSuite\Data”). It is possible to add your own replace-as-you-type combinations by creating a text file with the extension “.rayt” and saving it to this same folder. Such text files can be prepared with any basic text editor, such as Notepad. Unicode format text files are also supported. Each “replace-as-you-type” entry is entered on a new line simply as the string to be typed, followed by a comma, followed by the text to replace the string with, e.g. “(c),©”. After copying your file to the relevant data folder, restart the application for the changes to take effect. Any number of separate .rayt files may be placed in this folder; all will be loaded.

Inserting Phonetic Symbols (e.g. IPA)

The replace-as-you-type tool described above contains shortcuts for nearly all phonetic symbols. You could also configure keyboard shortcuts for these or other phonetic symbols.

Editing TLex "Smart Cross-references"

"Smart" Cross-references: Overview

When the user of a dictionary is referred from one place to another place within the central section of a dictionary, this is achieved by means of a cross-reference. For example, the treatment of the lemma "gray" may simply refer the user to the lemma "grey", where a full treatment of the lemma would be found:

gray = **GREY**
...
grey *(adjective)* of a colour that is intermediate between black and white; *(noun)* a neutral colour between black and white

A common problem in dictionaries is *invalid cross-references*. This is when the user is referred to a lemma that has *not been treated* in the dictionary. The user might also be referred to a *non-existent or incorrect sense*, or, in cases where the target lemma has multiple homonyms, the user may be referred to a *non-existent or incorrect homonym*. Another common problem is *ambiguous* cross-references, where the user is referred to a lemma, but then finds that that lemma has multiple homonyms (or senses), and it is not clear which homonym (or sense) was intended. Keeping track of cross-references in order to reduce or eliminate such errors is a huge task, if done manually. The TLex "smart" cross-references system, however, solves these problems. TLex allows you to enter cross-references in such a way that the system "knows" the actual structure of the cross-reference, i.e. internal "links" are established between the **reference *position*** (where one finds the cross-reference), the **reference *marker*** (i.e. the type of cross-reference), and the **reference *address*** (or target). This has a couple of important and powerful implications:

1. A cross-referenced lemma sign or sense (the reference address) *has to exist* – TLex thus *prevents* you from being able to create invalid cross-references to non-existent lemma signs or senses.
2. Because of the automatic numbering of homonyms and senses in TLex, the homonym and sense numbers of all cross-references to a lemma sign or sense at a reference position will be *automatically updated* whenever the homonym or sense number change on the target.
3. If the form (orthography) of the cross-referred lemma sign changes, the same change will be automatically effected at the reference position.

The TLex cross-references system thus frees you from having to manually keep track of cross-references, and also frees you from having to manually update the cross-reference homonym and sense numbers whenever you change the order of homonyms or senses of a cross-reference target. It also frees you from having to double-check whether the spelling at the reference position and the reference address are the same.

Further, the Preview Area also automatically displays all articles that are cross-referenced by the lemma you are currently working on, as well as all lemmas that have cross-references to the lemma you are currently working on. This allows you to quickly and continually check the correctness of cross-references at all times. This is shown in the screenshot below:

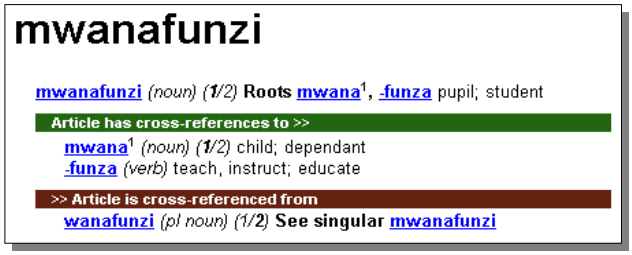


Figure 14: *Related cross-referenced articles 'to and from' are shown in the Preview Area, here for a bilingual Kiswahili – English dictionary [Data online at: http://africanlanguages.com/kdp/]*

In this example, the article for "mwanafunzi" contains two cross-references, one to "mwana" homonym number 1, and a second one to "-funza". The articles for both of these are shown, in that order, under the section labelled "Article has cross-references to >>". From the section with the heading ">> Article is cross-referenced from" one also sees that the article "wanafunzi" contains a cross-reference to "mwanafunzi". Especially the latter type is highly revealing. Further note that the reference *markers* with which the reference *relations* are being established ("Roots" and "See singular") are meaningful text segments. These markers are being selected from a drop down list when establishing the cross-reference relations, as shown in the next section.

Note that it is also possible to hide the outgoing and incoming cross-referenced articles, by toggling the status of the "Show related cross-references" checkbox under "Format (F4)".

Adding Cross-references

To create cross-references in TLex, one uses the special "References" element. Each "References" element can hold multiple cross-references. By default, the "References" element can be added either to "Lemma" elements, or to "Sense" elements (in cases where the cross-reference should only belong to particular word senses).

To add cross-references, right-click on the "Lemma" element or on a "Sense" element in the Tree View and select "Add: References". A "References" element will be added, and the cross-reference editor dialog (shown below) will appear. This dialog allows you to add cross-references to lemmas, or to particular senses within lemmas, by selecting them from a list of all the current lemmas and senses in the dictionary database.

It is thus important to realise that cross-references are never typed in manually. The reference position is defined by the place in the Tree View where one called up the cross-reference editor. The reference address(es) is/are chosen using the "Quick-search box", "Lemma list" and "Senses of selected lemma" sections of the cross-reference editor. The reference marker, finally, is chosen from the drop down menu indicated by "Cross-reference type selection".

When cross-references to several items are made, all are shown in the sub-window labelled "Cross-references". The order of these can be changed using the "Move up" and "Move down" buttons – see "Cross-reference order modification". A preview of the resulting article, together with the linked article(s) is shown in the "Cross-reference article preview".

If one wants to make a cross-reference to an item that has not as yet been treated in the dictionary, then one will typically create the lemma sign of that item first and mark the (empty) article as incomplete. This way the item becomes available in the cross-reference editor, and one can then fill in the contents at a later stage.

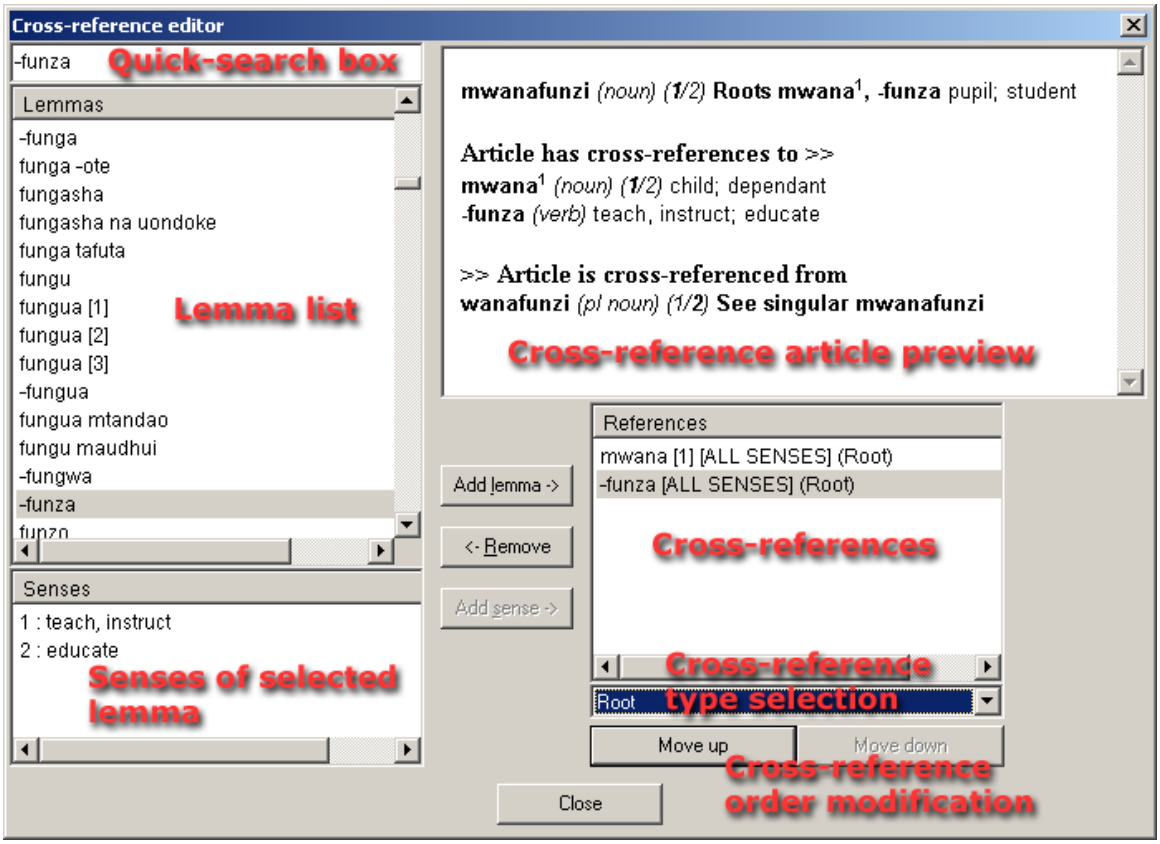


Figure 15: *Cross-reference editor dialog, here for a bilingual Kiswahili – English dictionary [Data online at: http://africanlanguages.com/kdp/]*

Editing Cross-references

If you want to modify the cross-references under an existing "References" element, you can do this by opening the Cross-reference Editor again for that element. There are two possible ways to do this:

1. Right-click on the "References" element in the Tree View and select "Edit".
2. Double-click on the "References" element in the Tree View.

Customising the Cross-reference Types

When you create a new dictionary in TLex, a default list of cross-reference types is set up, with, amongst others, "equivalence"/"=", "see" and "compare". It is possible to remove these or add new types, using the cross-reference types editor dialog accessible from "Dictionary/Edit cross-reference types". For more details on how to do this, see the chapter on Customising the cross-reference types.

Allowing Cross-references Between Sections in Multi-section Documents

By default, smart cross-references may link entries *within* the same section only. If you need cross-references *between different sections* for your project, this can be enabled by going to "Dictionary/Properties", then to the "Properties" tab, and enabling "**AllowRefsToAllSections**".

Search (F3)

Overview

The “Search (F3)” tool in TLex/tlTerm allows you to perform a full text search on the entire dictionary or terminology database.

Note that if a filter is currently applied, the search function will only search the subset of entries currently revealed by the filter (see the chapter on the “Filter (F5)” tool for more information).

Search Options

“Case Sensitive”

When the “Case sensitive” option is ticked, the search function will regard the case of a letter (i.e. uppercase/lowercase) as important while searching, and will not match results where the case differs from the search string.

“Whole Word Only”

When the “Whole word only” option is ticked, the search function will only return articles where the search string occurs as a whole word, and not those where the search string appears as part of a word. This difference is illustrated in the screenshots below:

Figure 16: *Searching with partial word matching in a bilingual English – Chinese dictionary*
[Data: © Lorraine Liang]

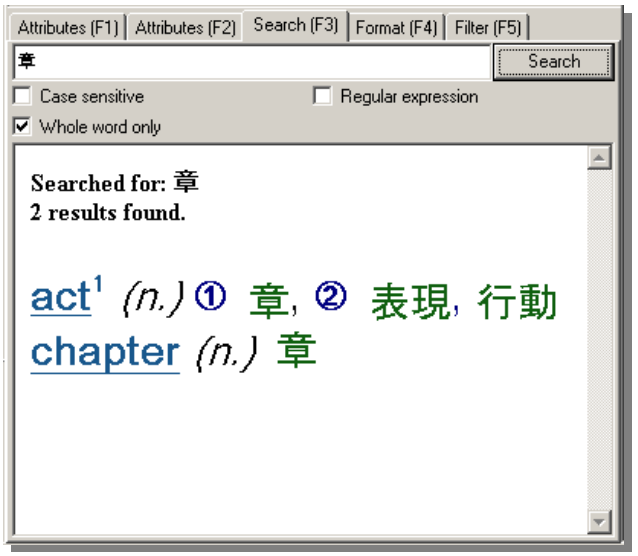


Figure 17: *Searching with “Whole word only” matching in a bilingual English – Chinese dictionary*
[Data: © Lorraine Liang]

“Regular Expression” Searches [Advanced]

Ticking the “Regular expression” option allows you to perform “regular expression” searches. This is a special “language”, with its own syntax, for creating more complex search queries.

The basic purpose of regular expressions is to allow you to create a single search string that can match *multiple* possible strings in the dictionary text. As a simple example, one can search for occurrences of either “gray” or “grey” (two alternative spellings of the same word) by using the search string “gray|grey”. The vertical line “|” is a regular expression construct that denotes “OR”.

Broadly speaking, regular expressions are an industry standard, and many detailed references are available on the World Wide Web, such as the following recommended Wikipedia article: http://en.wikipedia.org/wiki/Regular_expression. Thus, only a brief overview will be provided here. Note that there are different varieties of regular expressions; we use the so-called “Perl syntax”.

Following are a few more examples of regular expression constructs:

Single-character wildcard: “.” can be used to indicate “one of any character”, thus the search string “li.e”, will match “like”, “line”, “life”, “lime”, “lice”, etc. (but not “lie”).

Multi-character wildcard: “.*”

Beginning of field: “^” can be used to indicate “beginning of field”, thus allowing you to search for attributes that *start* with a given text string, e.g. “^to”.

End of field: “$” can be used to indicate “end of field”, thus allowing you to search for attributes that *end* with a given text string, e.g. “cat$”.

Empty field: “^$” (meaning, match beginning of field followed immediately by end of field)

One or more: “+” denotes that the preceding character may be repeated one or more times, e.g. “lo+se” will find occurrences of both “lose” and “loose”.

Zero or more: “*” denotes the the preceding character may be repeated zero or more times.

Character group: […], e.g. “gr[ea]y” will match both “grey” and “gray”.

Character range: [...-...], e.g. “[0-9]” will match any numerical digit from 0 to 9, and “[a-z]” will match any character in the 26-character English alphabet.

The regular expression search string “^[a-z][a-z][a-z]$” would thus, for example, find all attribute values that are just a single three-letter word.

Inverse (not): [^...], e.g. “[^a]” will match any character other than “a”, “[^]” will match any character other than space, and “[^aeiou]” will match any character that isn't a vowel.

Beginning-of-word: \<

End-of-word: \>

If you wish to actually search for one of the characters that are used as special regular expression query characters, you can do so by “escaping” the character – this is done by preceding it with a “\” character. Thus “\.$”, for example, will find all articles where an attribute actually ends with a “.”.

Numerous useful regular expressions for dictionary compilation work are given in the FAQ (Frequently Asked Questions – http://tshwanedje.com/faq.html), e.g. “Find all definitions or examples which end with a fullstop”.

A Few Useful Regular Expression Example Applications

Following is a small sample of ideas of tasks that can be achieved with the help of regular expressions.

Find all examples (or combinations) in which the "~" was not used

Do a field-specific (e.g. "Example::Example") regular expression search for "^[^~]+$".

Find all definitions (or examples etc.) which end with a fullstop

Do a field-specific (e.g. "Definition::Definition") regular expression search for "\.$".

Find all definitions (or examples etc.) which do not end with a fullstop

Do a field-specific (e.g. "Definition::Definition") regular expression search for "[^\.]$".

Find all definitions (or examples etc.) which begin with an uppercase letter (or with a lowercase letter)

Do a field-specific (e.g. "Definition::Definition") case-sensitive regular expression search for "^[A-Z]" (or "^[a-z]" for lowercase letters).

Find all single-word (i.e. non-multi-word) headwords/terms which begin with an uppercase letter

Do a case-sensitive regular expression search under F3 for "^[A-Z][^]*$".

(A mistake people sometimes make in terminology or dictionary work, particularly inexperienced users, is to enter headwords or terms starting with a capital letter, e.g. "Computer" instead of "computer"; this search can help automatically find such cases, while ignoring multi-word terms that should actually be uppercase, e.g. "World Wide Web".)

Find all two-word headwords

Do a field-specific (e.g. “Lemma::LemmaSign”) regular expression search for:

^[^]+ [^]+$

“Find” Tool

In addition to the search functionality available under F3, a full database “find” tool is accessible from the “Edit” menu. This can be invoked with “Ctrl+F”.

The “Find” tool includes the same basic options as Search (F3), namely “whole word only”, “case sensitive”, “regular expression”, and a “Fields” tab that allows you to restrict the search to particular attributes only. In addition to this, it also has the following four commands:

“Tag all”: Tag all matching entries. This adds to any current selection of tagged entries.
“Untag all”: Untag all matching entries. This does not clear existing tags from non-matching entries.
“Filter +”: Create and apply a “Reveal” filter that displays only the entries containing matches.
“Filter -”: Create and apply a “Hide” filter that displays all entries that do not contain matches.

See the relevant sections on tagging and filters for more information.

“Search and Replace” Tool

A full database “search and replace” tool is also accessible from the “Edit” menu. “Search and Replace” can be invoked with the shortcut key “Ctrl+H”. An alternate “Find” dialog can be invoked with “Ctrl+F”.

Additional advanced “search and replace” options allow you to do replacements without touching the user and date/time tracking fields - these should generally be used only if you know what you are doing.

Searching F2 Lists with “Find” and “Search and Replace”

You can search for multiple selections in an attribute by using a “,” (NB: without space) as the search string, e.g. “noun,verb” (NOT “noun, verb”). You can also do “intelligent” replaces; e.g. replacing “noun,verb” with just “noun” will clear the “verb” checkboxes. Replacing with “noun,pronoun” will clear the “verb” checkbox and tick the “pronoun” checkbox (providing it exists in the list).

Search “Smart Presets”

In both “Search (F3)” and the “Find” and “Replace” dialogs, a **“search presets” button (labelled “>>”)** allows you to select from various 'search pre-sets', with user-friendly names, that automatically fill in the corresponding (non-user-friendly) search query expressions and settings for you. These include:

(Last-modified/created date:)
Today
Yesterday
Last 2 days
Last 3 days
Last 7 days
Last 10 days
Last 14 days
Last 30 days
This month (so far)
Last two months
Last three months
Last month
-
Empty field
Non-empty field
-
Multi-word field
Single-word field

Two-word field
Three-word field
Four-word field
Five-word field
Two or more words
Three or more words
-
Starts with 'A'
Starts with uppercase
Starts with lowercase
Ends with fullstop
Ends not with fullstop
-
Reset settings

These can be helpful for particular tasks, e.g. the “Last 7 days” option can be used to help find and review all work done during the past week.

Format (F4)

In the sub-window “Format (F4)”, eight different types of general formatting settings may be modified. These settings affect how the output will appear in the Preview Area and when exporting the dictionary data (for printing or electronic display). In each case the Preview Area updates immediately as changes are made. In the screenshot below the settings have been chosen in such a way that they contrast the settings used in the screenshots so far, and/or in the screenshots shown in the following chapters.

Figure 18: *Manipulating general formatting settings under “Format (F4)”, here for a bilingual English – Chinese dictionary [Data: © Lorraine Liang]*

Alternative Sets of Labels for Lists

The largest section of the “Format (F4)” sub-window is reserved for “Alternative sets of labels for lists”. As will be explained in the section on Attribute Lists, in the chapter on Customising the Dictionary Grammar using the DTD, each attribute list can have any number of alternate/variant lists. In the screenshot above, two alternative sets for the parts of speech have been prepared, one in English, the other one in Chinese, with the Chinese version being chosen. Cf. the first black sections in italics and between brackets in the Preview Area.

Cross-reference Label Sets

Just as is the case for attribute lists, any number of alternate/variant sets of cross-references can be prepared, used and output. See the section on Customising the Language used to Display Cross-reference Types, in the chapter on Customising the Cross-reference Types. In the screenshot above,

the “Symbols” set has been chosen instead of the English/Chinese sets. As a result, the reference marker text “See” appears as a purple arrow “→”.

Show Related Cross-references [TLex]

In the “Smart” Cross-references section of the chapter on Editing Cross-references, it was pointed out that related cross-references departing *from* and arriving *at* the currently selected article are shown in the Preview Area, following the selected article. If one rather wishes to see the A to Z list only, then one can untick the “Show related cross-references” checkbox.

Expand Entities

Entities, discussed in the chapter on the Styles System, are automatically substituted with their associated values in the Preview Area when the “Expand entities” option is ticked. This is the default.

Show Frequency

When (corpus-derived) frequency data has been attached to the lemmas, this information can optionally be displayed. In the screenshot above, the rose-coloured values preceding the lemma signs are the rank numbers in the 100-million-word BNC (British National Corpus). (For more information on the latter, cf. Adam Kilgarriff’s “BNC database and word frequency lists” which is available from: http://www.kilgarriff.co.uk/bnc-readme.html)

Expand Tildes (~)

In the section titled Expansion of Tilde Character (~), in the chapter on Editing Attributes, the behaviour of the “~” was extensively discussed. Please refer to that section for more information. When the “Expand tildes (~)” checkbox is ticked, the tildes are automatically expanded.

Show Timestamp

The last-saved date and time for each article are automatically saved by TLex/tlTerm. This information can be revealed by ticking the “Show timestamp” option. Timestamps are displayed at the end of each article in the screenshot above.

Show Usernames

If 'User Management' is enabled on a dictionary or termbase, 'created-by user' and 'last-saved-by user' information is saved at each entry. This information can be revealed in the Preview Area by ticking the “Show usernames” option.

Find/Filter (F5)

The “Find/Filter (F5)” function (previously called “Filter (F5)” in older versions) allows you to display and work on only a *subset* of entries in the dictionary or terminology database, based on specified criteria. The filter acts as a kind of “mask” or “sieve” that “lets through” (‘filters’) only those entries that conform to the specified selected conditions.

The filter tool also acts as a kind of *search/find*, as it allows one to search for things in your data that match your selected conditions (e.g. ‘edited-by this user’, or ‘contains usage examples’).

Filters are very powerful, and useful for many purposes.

> **Tip:** While you have a filter *applied*, you can press Ctrl+G to jump to each next matching instance

There are many different possible uses for the filter. For example, one can select to “show only those dictionary entries which do not yet have usage examples”, or “show all dictionary entries that don't have a part of speech indicated”. Another example might be to “show all articles labelled as 'colloquial'”. A more complex example might be to “show all articles that are marked as complete and have definitions, but which lack a usage example or a part of speech label”. In a tlTerm termbase, the filter can be used to easily find all entries (concepts) that do not yet have a term filled in for a particular language, e.g. “Show me all entries that have English terms defined, but no Spanish equivalent”.

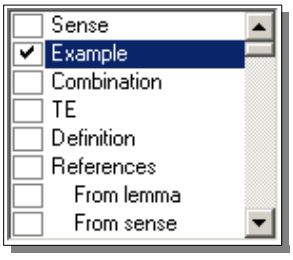
When a filter is applied, the Preview Area will display a message indicating it is, and show how many entries, out of the total number of entries, have passed the filter, e.g. “***[Filtered: 254 of 3289]***”.

Many other features are also “aware” of the filter, and may accordingly work only on the subset of entries revealed by the currently applied filter. These include the “Search (F3)” function, TLex's Ruler Tool, exporters and more.

> **Note:** Filters are temporary (i.e. by default not saved) and are cleared when you exit. In multi-user editing setups they are also *only* local to your computer.

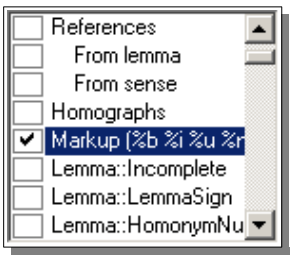
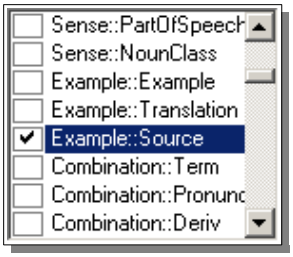
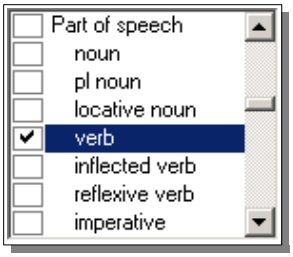
Types of Filter Conditions

There are several types of filter conditions. These are all listed under “Filter (F5)”. Each condition has a checkbox to the left of it, which may be ticked in order to select that condition. The main types of filter conditions are as follows:



Element Filter: Checks for the *presence* of the given element type within the tree structure of an article, e.g. “Sense”, “Combination”, or “Example”.

For the cross-references element (“References”), one may also further check for *only cross-references from a Lemma*, or *only cross-references from a Sense*.

	**Markup Filter:** Checks if any markup characters (e.g. "%b") appear within an article.
	**Attribute Filter:** Checks if the given attribute value *appears* in an article *and has a value filled in*. Attribute filter conditions are listed in the form "ElementName::AttributeName". For instance, ticking "Example::Source" would check for articles that contain an "Example" element *and* have a value filled in for the "Source" attribute of the "Example" element.
	**List Filter:** Checks if any list attributes in an article have the specified list item(s) selected. Note that ticking the list *name* (e.g. "Part of speech") instead of a particular item (e.g. "noun") will check if *any* of the list items (in this case for "Part of speech") are selected in an article.

Two sets of filter conditions may be specified – conditions for *inclusion* of entries ("Reveal"), and conditions for *exclusion* of entries ("Hide"). These may be combined to create more complex filters.

To apply a filter, press "F5" to select the Filter tool, specify the filter mode (e.g. "AND" or "OR"), tick the desired conditions, then click "Apply". To unapply the filter again, click "Unapply". To clear the currently ticked conditions and mode, click "Reset".

"Reveal" Filter

The "Reveal" filter (previously called the "Include" filter) is used to reveal only entries that *conform* to the specified conditions. By selecting between "Any of (OR)" and "All of (AND)", one can further specify whether only *one or more* of multiple specified conditions needs to be present, or whether multiple specified conditions should *all* be present, respectively, in order to include the entry. Some examples follow:

1. "Show only entries marked as incomplete"
- Under "Reveal", select "Any of (OR)"
- Tick "Lemma::Incomplete"
- Click "Apply"

2. "Show all homonyms"
- Under "Reveal", select "Any of (OR)"
- Tick "Lemma::HomonymNumber"
- Click "Apply"

3. "Show only nouns and verbs"
- Under "Reveal", select "Any of (OR)"
- Tick "noun" under "Part of speech"
- Tick "verb" under "Part of speech"

- Click “Apply”

The “OR” option is used here as the article may have either the “noun” label or the “verb” label selected for the lemma to be revealed by the filter.

4. “Show all nouns that belong to the 'linguistics' domain”

- Under “Reveal”, select “All of (AND)”
- Tick “noun” under “Part of speech”
- Tick “linguistics” under “Domain label”
- Click “Apply”

Note that this example assumes that you have a “linguistics” domain label. The “AND” option is used because *both* conditions must be satisfied for an article to be shown (i.e. lemma “must be a noun” AND “must have a linguistics label”).

“Hide” Filter

The “Hide” filter (previously called the “Subtract” filter) can be used to reveal only those entries that do *not* conform to specified conditions. Phrased alternatively: The “Hide” filter can be used to *hide* all entries that *conform* to the specified conditions. By selecting between “Any of (OR)” and “All of (AND)”, one can further specify whether only *one or more* of multiple specified conditions needs to be present, or whether multiple specified conditions must *all* be present, respectively, for an entry to be hidden by the filter. Some examples follow:

1. “Show all entries that do not have usage examples”

- Under “Hide”, select “Any of (OR)”
- In the conditions list, tick “Example”
- Click “Apply”

2. “Show all entries that do not have the part of speech specified”

- Under “Hide”, select “Any of (OR)”
- In the conditions list, tick “Part of speech”
- Click “Apply”

Combining “Reveal” and “Hide”

The “Reveal” and “Hide” filters may be combined, allowing more complex filters to be defined. Some examples follow:

1. “Show all incomplete articles that have definitions but not usage examples”

- Under “Reveal”, select “All of (AND)”
- In the “Reveal” conditions list, tick “Lemma::Incomplete”
- In the “Reveal” conditions list, tick “Definition”
- Under “Hide”, select “Any of (OR)”
- In the “Hide” conditions list, tick “Example”
- Click “Apply”

The “Reveal” part of this filter will firstly select only the subset of articles that are marked as incomplete and that have definitions; the “Hide” part of this filter will then subtract from that all those lemmas that have usage examples.

2. [Note: In Bantu languages such as Kiswahili, nouns are grouped in singular/plural classes.]
In the screenshot below, all “complete” nouns that have been marked with the singular “class 9” or the corresponding plural “class 10”, and are “animate”, but for which the treatment does not have an

"example sentence" or a "combination", are being extracted. Out of the 6,147 lemmas in the database, this condition is fulfilled for only 46 of them, as shown at the top of the Preview Area.

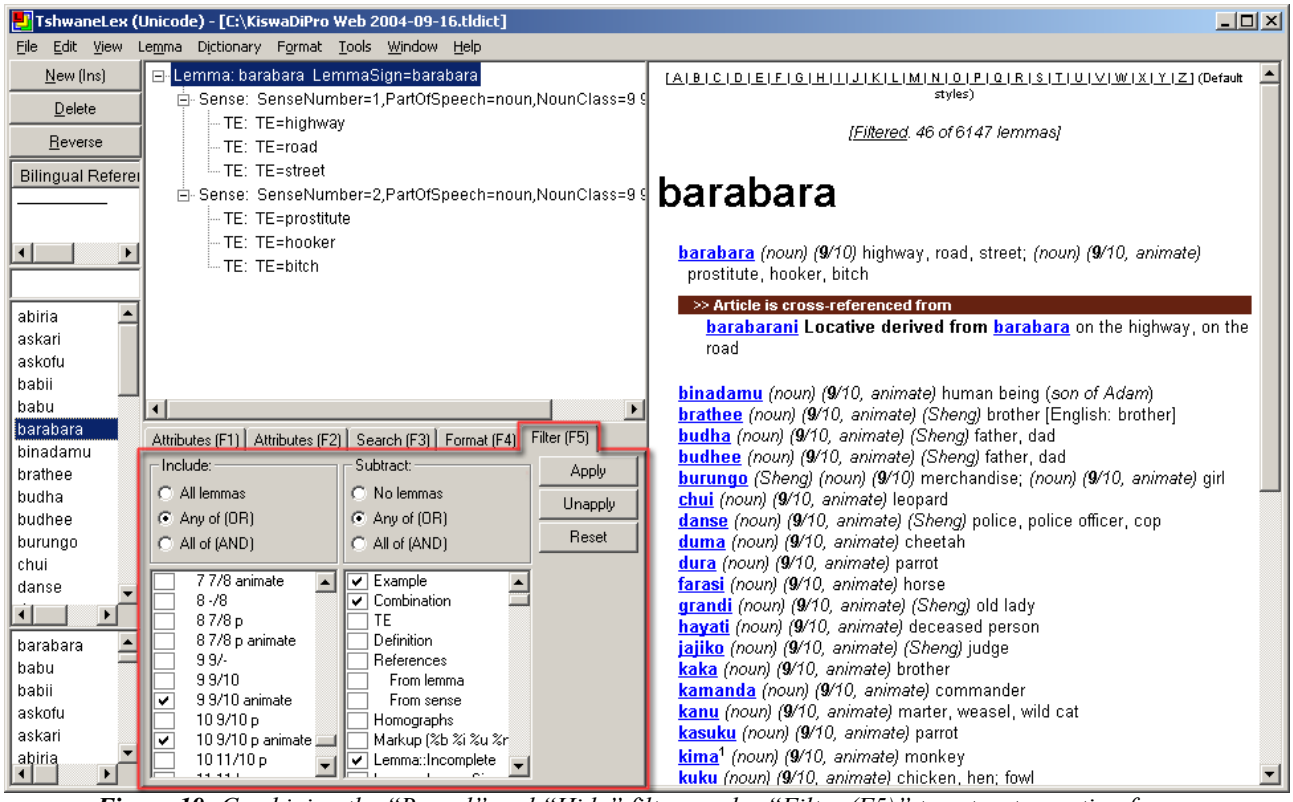


Figure 19: *Combining the "Reveal" and "Hide" filters under "Filter (F5)" to extract a section from a bilingual Swahili – English dictionary [Data online at: http://africanlanguages.com/swahili/]*

Extracting or Printing a Subset of the Dictionary/Termbase

The filter may be used to select and export only a subset of the dictionary or termbase. When you export your data (whether to RTF, HTML or XML format), the "Export options" dialog includes an option to "Use filters". Selecting this option will cause the current filter to be applied to the output.

As an example, one could choose to export only the lemmas in a dictionary that have been marked with a "linguistics" label, effectively generating a linguistics lexicon or terminology list from the main dictionary database.

By combining the filter with the use of multiple sets of styles, which may further hide or reveal different elements or attributes, multiple dictionaries can effectively be generated from a single dictionary database. See the chapter on the Styles System for more information.

Corpus (F6)

Configuring the Corpus

Step-by-step:

1. Prepare your corpus files as text files. (It may be a good idea to save them all to a specific dedicated folder, but that is not necessary.)
2. Under "Corpus (F6)", click on "Configure" and select "Texts/Add multiple".
3. Click "Browse" to select the folder containing the text files (the "recurse" option will specify whether or not TLex/tlTerm will also auto-add text files from subfolders within the selected folder).
4. Click "OK".
5. The desired corpus files should now appear in the list.

The configuration will be saved along with the particular database.

More files can be added at any time later, or files may be removed from the list.

The “right to left” checkbox allows you to specify if a corpus file contains text in a right-to-left language, such as Arabic or Hebrew.

Doing a Corpus Search

Once the corpus files are configured, you can perform search queries on the corpus. Either a query can be entered manually under "Corpus (F6)", or you can tick the "Auto-search" option, and TLex/tlTerm will then automatically launch a corpus search for the current headword/term each time you select an entry.

The most recent results are kept in memory, thus if you select another entry and then go back again to the first entry, the search results should re-appear immediately. If the search had not yet completed, it will automatically continue on its way again.

Sorting the Results

The ordering of corpus search results can be configured by clicking on "Configure" under "Corpus (F6)", and under "Sort", using "Move up" and "Move down" to change the order of sort items. For example, by moving "Word Before Search Term" to the first position, the entries will first be sorted on the word to the left of the search term within a results line. (If the word to the left is the same for two lines, the next item in the list will decide how they are further sorted, and so on.)

Auto-grabbing Usage Examples

One of the powerful time-saving features of TLex/tlTerm is the ability to automatically 'grab' a sentence from a line in the corpus results and attach it as a usage example in the current entry. To do this, use the following procedure:

1. Select the desired 'Sense' element in the Tree View to which you wish to attach the example
2. Select the desired line (or lines) in the corpus results by clicking on its number in the left column
3. Press the shortcut key **Ctrl+F7**

Note: This "relies on" the default "Sense" and "Example" elements from the default TLex DTD being present.

It is also possible to grab multiple sentences at a time. Just select the desired corpus lines (e.g. holding in "Ctrl" while clicking on them with the mouse to create a multi-selection) and press Ctrl+F7; each one will be added as an 'Example' to the currently selected 'Sense'.

Copying Selected Examples (Corpus Lines) to the Clipboard

You can use the shortcut "Ctrl+C" to copy the currently selected corpus line to the clipboard.

Corpus Encryption

The corpus (F6) tool includes a facility to 'encrypt' corpus files and protect them with a password. The resultant encrypted files can be used *only* within the "Corpus (F6)" tool, but outside of TLex/tlTerm the files will be unreadable. This allows you to protect your corpus from potential theft by members of your team or anyone else with access to their computers.

To apply encryption to all or part of your corpus, click on "Configure" under "Corpus (F6)", click on the "Texts" tab, then select one or more files that you would like to encrypt from the list. Multiple files can be selected by holding in "Ctrl" on the keyboard while clicking with the mouse. Alternatively, if you wish to encrypt all files, click on "Encrypt all". You will be prompted for the password that will be used to protect the files. Enter the password carefully and click "OK". New copies of each chosen file will be saved (to the same folder) with an extension ".tecrypt". These files may then henceforth be distributed to the compilers instead of the original text files.

IMPORTANT:

- Make sure to keep a backup copy of the original corpus files in a safe place. Do not lose the originals. If you forget the password, the original files can not be recovered.
- The encryption password is case-sensitive, meaning "a" is considered different from "A".

Corpus Registry Settings [Advanced]

Increasing the Maximum Number of Corpus Lines Returned

By default only the first 1000 results for a search are returned; this can be changed. Under "HKEY_CURRENT_USER/Software/TshwaneDJe/(ApplicationName)/Settings" (create if necessary), create a DWORD registry key called "MaxCorpusResults" and set the value to the desired maximum number of corpus lines.

Increasing the Maximum Number of Cached Sets of Search Results

The most recent sets of corpus search results are cached, so that if you immediately return to a recent search, the results can be displayed immediately. The number of results to cache can be changed. Under "HKEY_CURRENT_USER/Software/TshwaneDJe/(ApplicationName)/Settings" (create if necessary), create a DWORD registry key called "MaxCorpusCachedResults" and set the value to the desired maximum number of search results. The default is 20.

CAUTION: Changing this to a very high number may impact performance.

Tags

Overview

Entries in your data can be “tagged”. This is a temporary 'flag' on an entry that merely marks that entry as being tagged, allowing you to do certain things with it, such as filter on all tagged entries (see the section on filters for more information). Tagged entries display in the Preview Area with a red half-border to the left and top, and with a small image of a tag next to them.

Tags are not saved with your document, and are cleared when you close it.

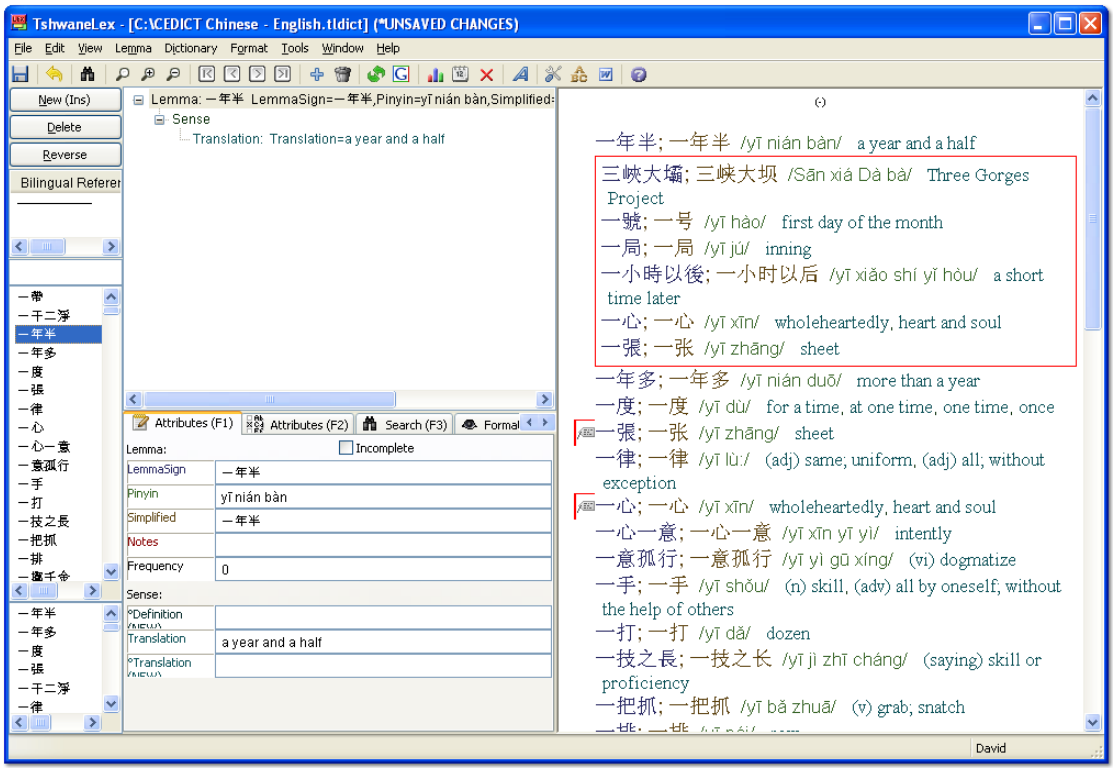


Figure 20: Tagged entries in a Chinese - English dictionary in TLex. The "Show tagged always" option is enabled, thereby showing all tagged entries with a red border in the Preview Area at all times. (The data is from the CEDICT project and is online at http://dictionaryq.com/chinese/.)

Tagging Commands

“Edit/Tag entry” (Ctrl+F2): Tag or untag the currently selected entry.

“Edit/Tag all”: Tag all entries in the Lemma/Term List (of the current language/section). This is “filter-aware” - if a filter is currently applied, only the entries that pass the filter will be tagged. This allows for a new way to *combine* the results of several different filters – by tagging the matching entries of each filter, one can then finally filter on “tagged” entries.

“Edit/Clear all tags”: Clear all current tags.

“Edit/Filter tagged”: Create and apply a filter that displays only currently tagged entries.

“Edit/Show tagged always”: Display all tagged entries in the Preview Area persistently. One use for this would be if you want to continually view or refer to an entry while working on another entry.

Tag Filter

Under “Filter (F5)”, the “[Tagged]” filter condition allows you to apply either a “reveal” or “hide” filter for tagged entries.

Tagging in the “Find” Tool

In addition to the “Edit” menu commands for tagging, the “Find” tool allows you to automatically tag or untag all entries that match a given search query.

This could be useful if you would like to combine the results of multiple different search queries, and e.g. create a filter from that.

Tagging a Range of Entries (e.g. an Alphabetic Section)

To tag any range of entries:

1. Select the first desired entry of the range in the entry list
2. Click “Edit/Tag entry”
3. Select the last desired entry of the range in the entry list
4. Click “Edit/Tag range”

If combined with a tag filter, this can then also be used to easily export a specific range of entries (e.g. a single alphabetic section) by selecting the 'Use filters' option at export time.

Microsoft Word/Excel Integration ('Search Widget')

TLex and tlTerm feature Microsoft Word/Excel Integration. This can be enabled or disabled via the "Tools/Microsoft Word integration" menu option, or via a toolbar button. (It can be also be hidden by clicking "Close" on the pop-up window itself, when open.)

Microsoft Word/Excel 'Search Widget'

When Microsoft Word Integration is enabled, TLex/tlTerm automatically do a search in the currently open database for the word on which the cursor is standing in Microsoft Word (and/or Excel, if open), and display a pop-up window in the lower right corner with a miniature version of the results of the search, if a corresponding entry is found. If the word is not found, a clickable option allows you to immediately **add** the word. This may be useful for lexicographers, and is ideal for terminographers. The pop-up window is also ideal for translators.

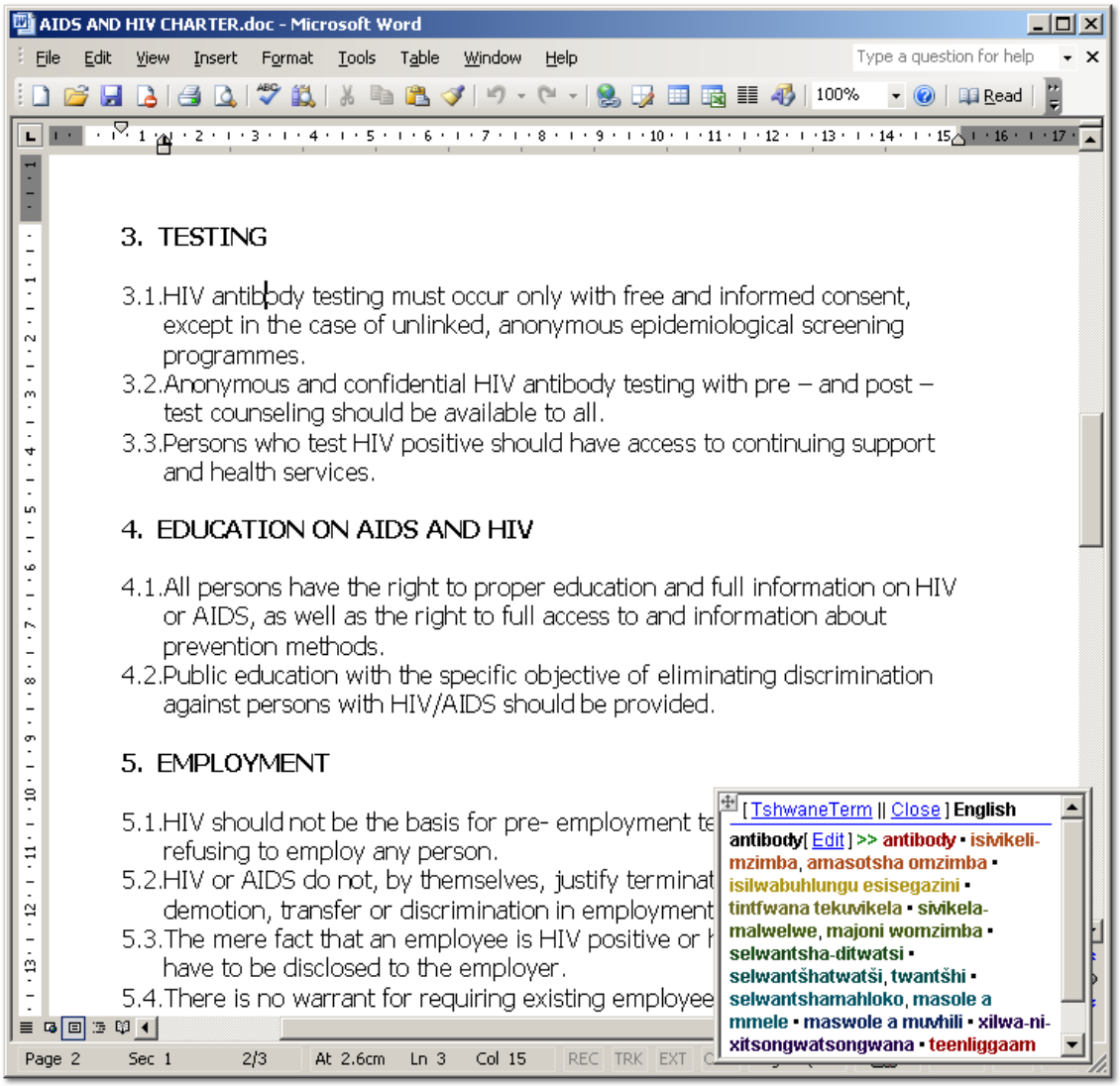


Figure 21: Microsoft Word Integration, which shows instant look-up results in your termbase or dictionary for the current word in Microsoft Word

Note that this functionality is also available to end-users in the TLex Electronic Dictionary/Termbase System.

Sending/Adding Text from TLex/tlTerm to Word/Excel

General 'Text Commands' Menu

Many parts of the user interface of the software display a "**Text-commands for ...**" menu, which contains general commands for the text you've clicked on. For example, if you right-click on the Entry List on the word "dog", you'll see this 'Text-commands menu' to perform various actions *for the word* "dog". Likewise, if you right-click on any term, translation or headword in the Tree View (and also when right-clicking on words in search results in tlCorpus).

This useful and powerful sub-menu contains many useful time-saving commands for the given text, e.g. to quickly launch a Web search or image search, to open "Character Information Tool", to do 'add as new entry', to 'copy to clipboard', to start an F3 search, to start an F6 corpus search, to start a tlCorpus search in tlCorpus linked mode, or to start a search in the Multi-Document Smart-Search window.

This sub-menu also contains the commands "**Insert into Word**" and "**Insert into Excel**". Activating these will 'send' the relevant text you've clicked on to Word or Excel respectively (if open), to where your editing cursor stands in Word or Excel.

This "Text-commands" sub-menu is also available from the 'Multi-Document Smart-Search' window in the search results, as well *within* search results showing in the Word/Office 'search widget' pop-up itself.

Video briefly showing the 'General Text Commands Menu':

https://youtu.be/Sj4fb6lGkGc?list=PL_xDVMreHQV13lJiTx6h0L04j_OiTUcDv&t=410

General Text Commands Menu in tlCorpus, and Time-saving Linked Mode

tlCorpus also has its own 'General Text Commands Menu', which is similar to that in TLex/tlTerm. This is available in various parts of tlCorpus, including when right-clicking on any word within the search results (*NB, only when the tlCorpus 'search results colours' option is enabled*).

When running tlCorpus in **Linked Mode**, there are some further useful time-saving commands to directly "**Insert as new entry in tlTerm/TLex/tlDatabase**" for words you right-click on.

This can be enormously time-saving for lexicography or terminology projects, as while viewing corpus results, one can very quickly add words you come across as new entries with just a few clicks.

See the chapter on Linked Mode for more information.

TLex Bilingual Editing Features

Linked View

When Linked View mode – which is available from the "View/Linked view" menu option – is enabled, related lemmas on the other side of the dictionary are immediately shown whenever you work on a lemma. That is, the language editing window for the *other* side automatically displays all articles whose lemma signs appear as a translation equivalent in the currently selected article. This allows for quick comparison, helping you to ensure balanced treatment of related lemmas on both sides of the dictionary.

In the screenshot shown in Figure 6, for example, work is being done on the isiNdebele to English side (cf. the red border around that language editing window), *while in Linked View mode*. The translation equivalents for the verb "-akhiwa" are "build" and "construct", which results in the automatic display of all entries on the English side where "build" and "construct" are lemma signs. For the verbal uses on the English side, one finds the translation equivalent "-akha" (the root of "-akhiwa"), which means that this dictionary is indeed balanced in this case. The reversibility principle has in other words been honoured here.

[The **reversibility principle** is the condition whereby all lexical items presented as lemma signs or translation equivalents in the X-Y section of a bilingual dictionary are respectively translation equivalents and lemma signs in the Y-X section of that dictionary.]

Automatic Lemma Reversal

TLex provides functions for automated reversal of articles. These functions are intended primarily as an aid to speeding up the compilation of the reverse side of the dictionary.

Single Article Reversal

To reverse the currently selected lemma, click on the "Reverse" button in the top left of the language editing window, or select the "Lemma/Auto-reverse lemma" menu option. The "Reverse tool" dialog will appear. This lists potential candidates for reversal – typically one per translation equivalent within the current article. In the left column of this window, you can toggle which of the translation equivalents you would like to generate reverse entries for by clicking on the corresponding "Included" option. Once you are satisfied with the selection, click the "Reverse" button. The selected new reversed lemmas will be generated and added to the other side of the dictionary.

The automatically generated entries will be marked as "incomplete", as they will typically require further work from you before they can be considered complete.

Full Dictionary Reversal

In cases where only one side of a dictionary has been treated, it may be desirable to "kick-start" the other side of the dictionary by performing a *full* language reversal. This can be done using the "Dictionary/Full language reversal" menu option. A confirmation dialog will be displayed; note that a full reversal cannot be undone, therefore it is strongly recommended that you save your current

work first and make a backup (“File/Create a backup”) before proceeding. Once you confirm that you wish to continue, the full language reversal options window will be displayed. This allows you to select whether or not to reverse single-word translation equivalents, or up to 2- and 3-word multi-word translation equivalents, or to simply reverse all translation equivalents (and paraphrases).

You may also choose to mark automatically reversed lemmas as “incomplete”, since automatically reversed lemmas will typically require further work and checking before they can be considered complete.

Translation Equivalent / Term Fanouts [TLex & tlTerm]

The “fanouts” tool can be enabled via the “View/Show fanouts” menu option in TLex, as well as tlTerm. This feature automatically shows entries related to the current one via a shared Translation Equivalent or term (e.g. all other entries in the *same side* that share a Translation Equivalent that also appears within the current entry). In many cases, the results are somewhat akin to a “thesaurus view”.

Compare/Merge

Overview

The Compare/Merge tool allows you to visually compare two dictionary databases, or two versions of a dictionary database, side by side. When there are differences, four options are provided for resolving those differences, namely “Add”, “Merge”, “Replace” and “Delete left”. This tool thus allows you to integrate changes made by other users back into the main dictionary database, or to integrate changes made on another computer (e.g., if you have worked at home, and now wish to add the changes made at home to the database at the office). It also allows you to merely check on the progress that has been made since a previous version of the dictionary, or by another lexicographer. The Compare/Merge tool can be accessed via the “File/Compare/merge dictionary” menu option.

As an example, assume that you usually work on the main TLex file for a Kiswahili – English dictionary, called “Kiswahili.tldict”, while your colleague, Sarah, works on her own copy of this database, called “Sarah.tldict”. Once a week, Sarah e-mails you her file, and you merge her changes into the main dictionary database. You then send her a copy of the new, up-to-date combined database, and she proceeds with her work on that file. In order to use the Compare/Merge tool to achieve this, you would do the following:

- Open your main dictionary file “Kiswahili.tldict” as you would usually do, e.g. using “File/Open: TLex file” and selecting it. (Always open the *main* dictionary database file first, and *then* the file you want to compare against and whose changes you want to integrate.)
- Select the “File/Compare/merge dictionary” menu option.
- A file selection dialog will appear, labelled “Choose a file to compare against”. Select “Sarah.tldict” and click “Open”.
- The Compare/Merge dialog will appear, as shown in Figure 22.

The Compare/Merge dialog is split into two halves – the left half displays a list of lemmas in the *main database* “Kiswahili.tldict” that was opened first; the right half displays a list of lemmas in the database you are comparing *against*, namely Sarah.tldict. The filenames are indicated in the headings above each list of lemmas. It is important to note that any changes that you make using the Compare/Merge dialog will always be made to the *main* (i.e. left) database.

The title bar of the dialog displays a summary of the number of differences found. There are four possible cases, each of which is displayed in a different colour in the list of lemmas:

1. **“Same”**: The lemma exists in both dictionary documents, and is identical in both, i.e. no changes have been made. These are displayed in **black** text with a white background (see e.g. “pombe”, “profesa” or “-puana” in the screenshot).
2. **“Different”** The lemma exists in both dictionary documents, but the article is different, i.e. changes have been made. These are displayed in either **purple** or **red** (see e.g. “polisi [1]”, respectively “-punguza”, in the screenshot). If the lemma in the compared-against (right) side has a newer “last-edited” timestamp, then purple is used. Otherwise, red is used.
3. **“Only left”:** The lemma exists in the main (left) dictionary document, but does not exist at all in the compared-against (right) dictionary document. In this example, this would imply that either you have added a new lemma to the Kiswahili.tldict file, or Sarah has deleted an existing lemma from her Sarah.tldict file. These are displayed in **blue** (see e.g. “polisi [2]” in the screenshot).
4. **“Only right”:** The lemma does not exist in the main (left) dictionary document, but does exist in the compared-against (right) dictionary document. In this example, this would imply that either

you have deleted an existing lemma from the Kiswahili.tldict file, or Sarah has added a new lemma to her Sarah.tldict file. These are displayed in **green** (see e.g. "-pokea", "polepole" or "-pona" in the screenshot).

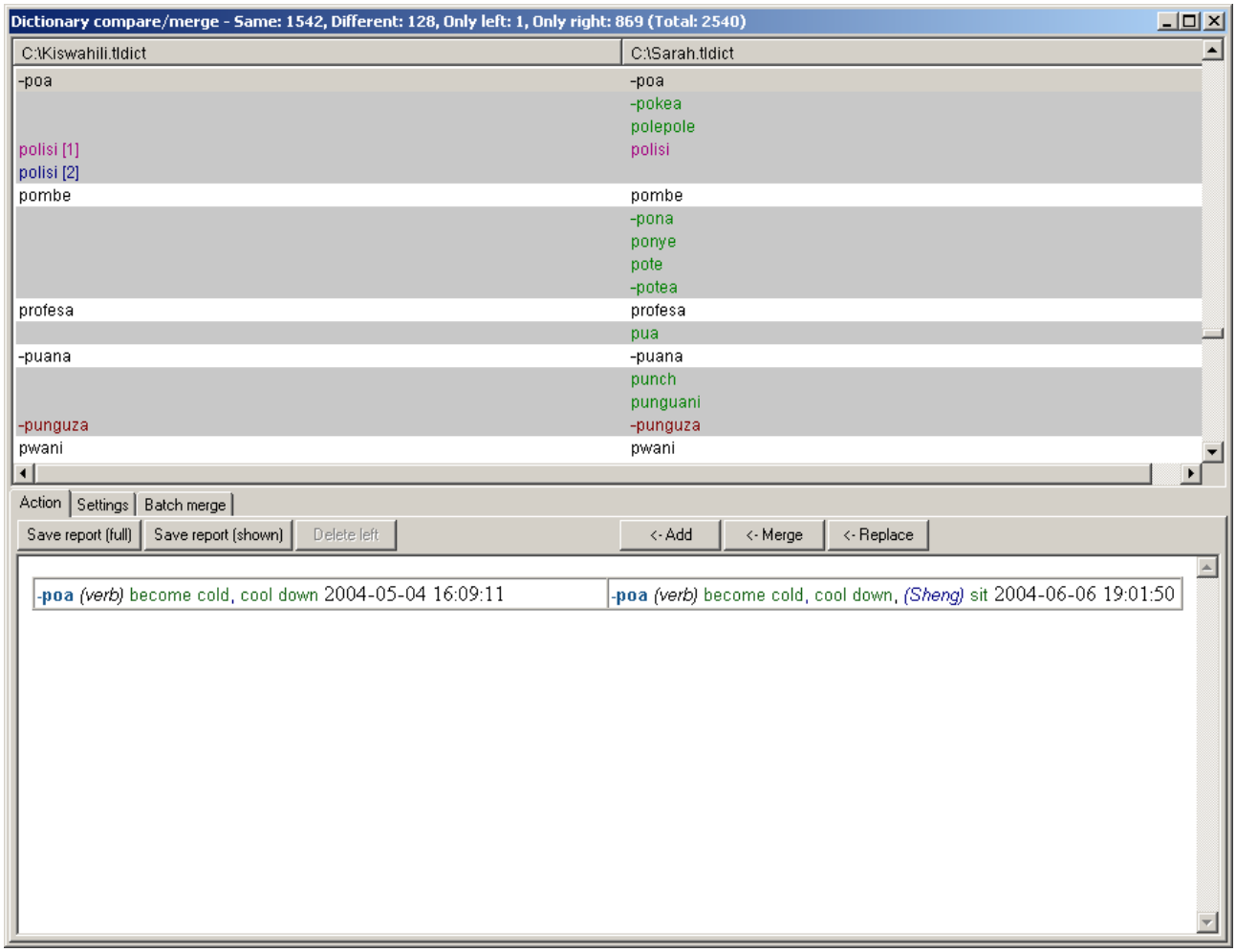


Figure 22: *"Action" tab in the dictionary Compare/Merge dialog, here comparing/merging two versions of a bilingual Kiswahili – English dictionary database [Data online at: http://africanlanguages.com/kdp/]*

Action Commands: "Add", "Merge", "Replace" and "Delete Left"

At this point you might then want to inspect Sarah's changes in each case where there are differences, and use the "Add", "Merge", "Replace" or "Delete left" commands, which are available on the "Action" tab, to then transfer her changes to the main database. Keep in mind that these commands will always make changes to Kiswahili.tldict (i.e. the file opened first), and never to Sarah.tldict. The four commands work as follows:

Add: The article on the right is *added* to the main database, in its entirety, as a new lemma. Use this if the lemma is a new one or a new homonym added by Sarah.

Merge: All senses and other child elements of the article on the right are *added* to the existing article on the left (i.e. the existing article in the main database). This option only applies to lemmas that exist in both databases, but are different.

Replace: The version of the article on the right *replaces* the version of the article on the left (i.e. the existing article in the main database). This option only applies to articles that exist in both

databases, but are different. Use this if Sarah has made *corrections or extensions* to an existing lemma.

Delete left: The article in the main (left) database is deleted. This option only applies to articles that exist in the main database only, i.e. those shown in blue. Use this if Sarah has deleted a lemma, and that lemma should also be deleted from the main database.

Note that it is possible to make changes to the main database while the Compare/Merge dialog is open. One may wish to resize the Compare/Merge dialog so that one can easily move from that window to the main dictionary database.

Settings

Typically one would not be interested in seeing, say, articles that are identical in both databases. It is possible to choose which of the four possible types of differences to display in the list. This can be done by clicking on the "Settings" tab, and using the four checkbox options, as shown below:

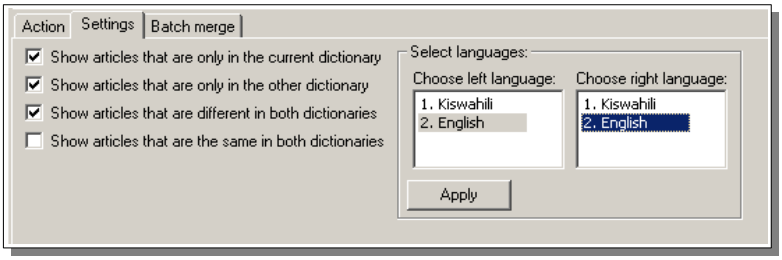


Figure 23: *"Settings" tab in the dictionary Compare/Merge dialog, here comparing/merging two versions of a bilingual Kiswahili – English dictionary database [Data online at: http://africanlanguages.com/kdp/]*

Bilingual Dictionaries: Choosing Which Sides to Compare

When comparing bilingual dictionary databases, by default the *left* sides of the dictionaries will be compared with one another. In order to select the other sides of the dictionaries, click on the "Settings" tab, select the desired sides to compare under "Select languages", and click "Apply". This is shown in the screenshot above.

Note that it is also possible to compare different sides/languages. This function may for example be used to reveal all lemma signs that have the same orthographic form across the two languages. Typically, one will not want to *merge* different sides however.

Batch Merge

If there are many changes in the compared-against database that you wish to merge into the main database, this can be done automatically, rather than adding/merging/replacing entries "one by one". This function thus allows you to add updates to the main database "in bulk" from the compared-against dictionary. To do this, select the "Batch merge" tab, as shown in the screenshot below:

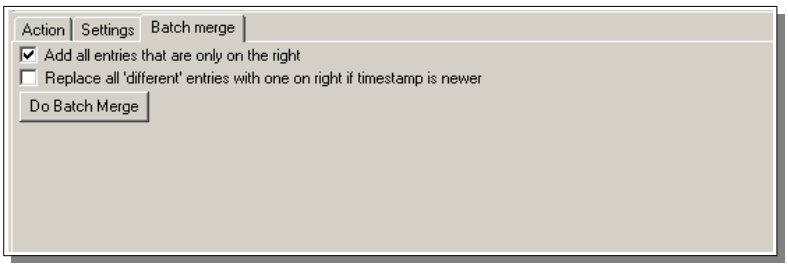


Figure 24: *"Batch merge" tab in the dictionary Compare/Merge dialog*

There are two options for "Batch merge":

1. **"Add all entries that are only on the right"**: When this is selected, all articles that appear in the compared-against (right) dictionary document but not in the main (left) dictionary document will be added during the "batch merge". These are all the entries marked in **green**.
2. **"Replace all 'different' entries with one on right if timestamp is newer"**: When this is selected, all articles that appear in both databases, but where the version on the right is *newer*, will be *replaced* by the newer version. These are all the entries marked in **purple**.

Once you have selected the desired options, click "Do Batch Merge" to go ahead with the merge.

⚠ **Important:** It is strongly recommended that you make a **backup** of the main database ("File/Create a backup") before doing a "batch merge".

[Advanced] Comparing DTDs (Document Type Definition / Field Descriptors)

Occasionally it may be useful to compare the DTDs of two different documents, or between two versions of your data. (The DTD is the "Document Type Definition" that defines the allowed fields in your document, e.g. elements and attributes, as well as some associated information like styles – see the chapter on DTDs for more information.)

The "Compare/Merge" tool contains a "**Compare DTDs**" function that can be used to compare DTDs. To use it, first open the two documents to be compared as per a usual Compare/Merge workflow explained in this chapter, then go to the "Compare/Merge DTD" tab, then click "Compare DTDs".

Note: This tool does *not* compare DTDs that are in the official XML ".dtd" format - it just compares the software's own internal DTD structures in-memory within this application.

Merging DTDs

You may also attempt to *merge* or combine two different DTDs by using the "Merge DTD" command, under the "Compare/Merge DTDs" tab. This will attempt to add e.g. elements and attributes that are 'only on the right' to the DTD of the main document, as well as styling information and output order information.

NOTE: Merging DTDs can be a finicky and imperfect process – it is not a 'magic button' that solves e.g. problems stemming from poor DTD management workflows, and might not get exactly your desired results if not used carefully – it is more intended to just help simplify combining e.g. small incremental DTD changes between versions of your database/document, within proper DTD management workflow practices. It may also be used to help set up the desired fields when

importing new sets of data that may be similar to other documents you've worked with. Use with care, and check the results carefully after running it.

As with the rest of the Compare/Merge tool, DTD merges happen from 'right to left'.

⚠ **Important:** You must make backups of your data before using this.

Merging User Lists

You can merge the list of user logins from another document into your document by using the "Merge users" command, under the "Compare/Merge DTDs" tab. (See the section on User Management for more information on user lists.)

This may be useful in cases like:
(1) Setting up a new project where you want to start with the same user logins
(2) If you've added new user logins in one copy of a file that contains some changes that should be merged with another copy, and you need the new user logins in the other copy too.

⚠ **Important:** You should make backups of your data before using this.

⚠ **Important:** When merging entries from right to left, information like "last-modified-by" and "created-by" requires that the usernames and/or IDs be present in the main (left) document. Some discrepancies may occur if user names or IDs have changed or do not match between the two documents (e.g. when merging user lists, the system cannot create two users with the same *internal ID*, but the "last-modified-by" information is saved under the *user ID*). So as always, you should be careful and always try to follow good workflow practices.

If you allow poor workflow management to get out of hand, it may become difficult to resolve this nicely (e.g. if you have several 'branches' or 'copies' of a document, and new users have been added separately in each, then the system can't perfectly merge these into a single combined user list, because the different new users may have the same internal IDs across different copies).

Ruler Tool [Advanced]

What is a Multidimensional Lexicographic Ruler?

It stands to reason that the various alphabetical categories in dictionaries have different relative sizes compared to one another. In English, for example, the alphabetical category **S** is huge compared to, say, **X**, **Y** and **Z**. When one compares existing dictionaries for a particular language with one another, one notices a remarkable consistency: expressed as a percentage, the size of a certain alphabetical category is always roughly the same. Moreover, when one compares an entire alphabetical breakdown derived from existing dictionaries, with an alphabetical breakdown derived from electronic corpora of that language, one notices that the two distributions correlate to a very high degree. It is, in other words, thus more than feasible to design a tool for each language with witch one can guide the compilation of new dictionaries. In the lexicographic literature, that tool has become known as a "multidimensional lexicographic Ruler".

In its most basic form a Ruler is an *abstract entity* since a single series of percentages straightforwardly correlates with the relative alphabetical allocation in **semasiological dictionaries**(i.e. dictionaries that are arranged alphabetically). As such, each alphabetical category is assigned a certain percentage, reflecting the relative size of that category. Different languages, and even different types of dictionaries for a specific language, have different Rulers. A general-language Ruler for a particular language, however, is valid for all general-language dictionaries of that particular language, and is also stable across time. Just as physical rulers with which one measures, Rulers can be made as fine-grained as one wishes by simply breaking down the alphabetical categories into smaller sections. Just as the human rulers who govern us, a Ruler can be called in to manage a project. Reformulated, a Ruler is a powerful instrument with which *measurements* and *predictions* can be made on various macro- and microstructural dictionary levels. These levels are meta-levels since one is not taking any individual semantics (e.g. the number of senses) or individual grammar (e.g. parts of speech) of single articles into account; one rather works with *averages* across a dictionary. It is precisely because averages are used that the tool crystallised into a single Ruler valid on multiple dictionary levels, hence its characterisation as a multidimensional lexicographic Ruler.

Rulers in TLex

In TLex, some advanced functionality has been implemented in order to work with Rulers in a user-friendly way. For example, if one wants to compile a new English – isiNdebele – English dictionary, then one can load the so-called "Ruler data" (i.e. the relative allocations to each alphabetical category, expressed in percent) for these two languages. Rulers for both languages have been designed, and the data can be found in the lexicographic literature.

The first screenshot below, Figure 25, shows the "Configure Rulers" tab, accessible from the "Dictionary/Ruler tool" menu option. The percentages one sees are the breakdown for English general-language reference works. According to this Ruler, the English alphabetical category **S** makes up 11.23% of the total, while **Z** only covers 0.27%.

These percentages can also be displayed as a coloured strip, one colour per alphabetical category, as seen in the first (top) Ruler in Figure 26.

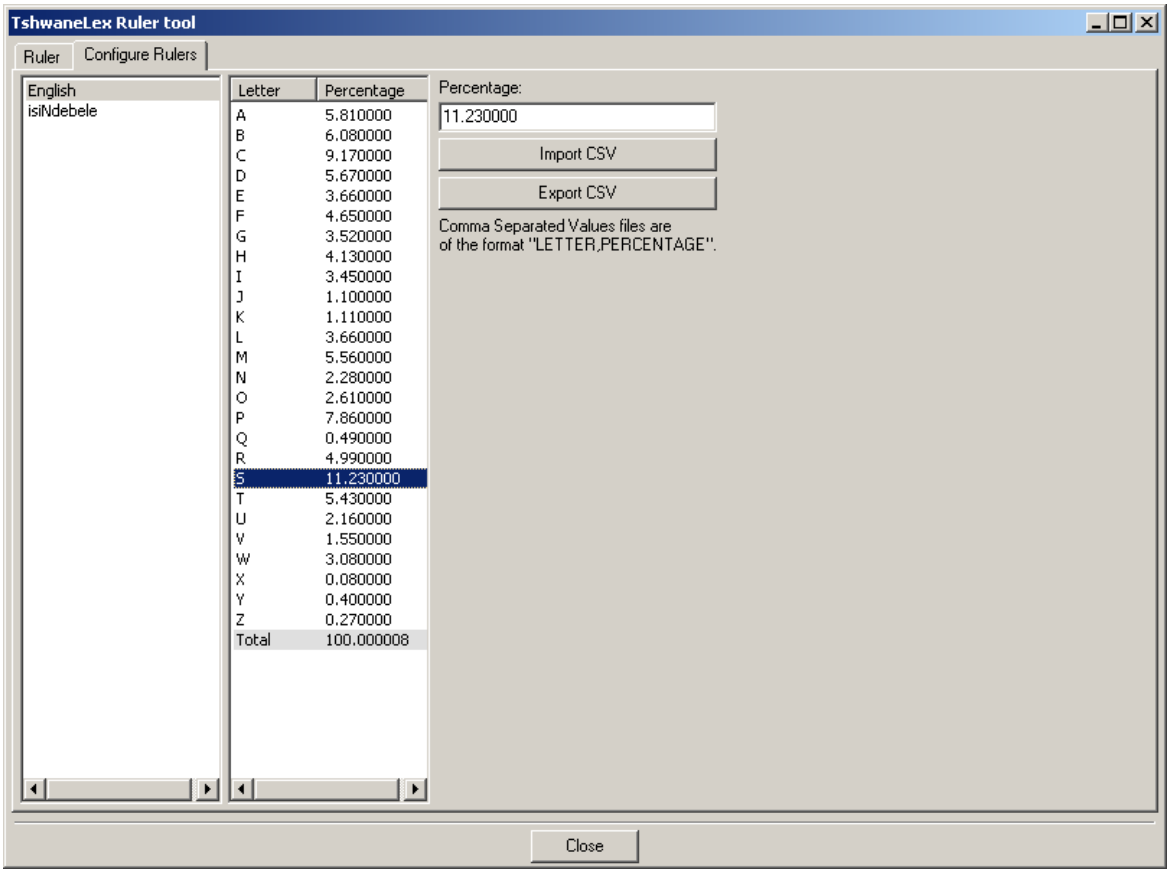


Figure 25: *TLex Ruler tool dialog, Configure Rulers tab,*
here configured for a general-language English dictionary side

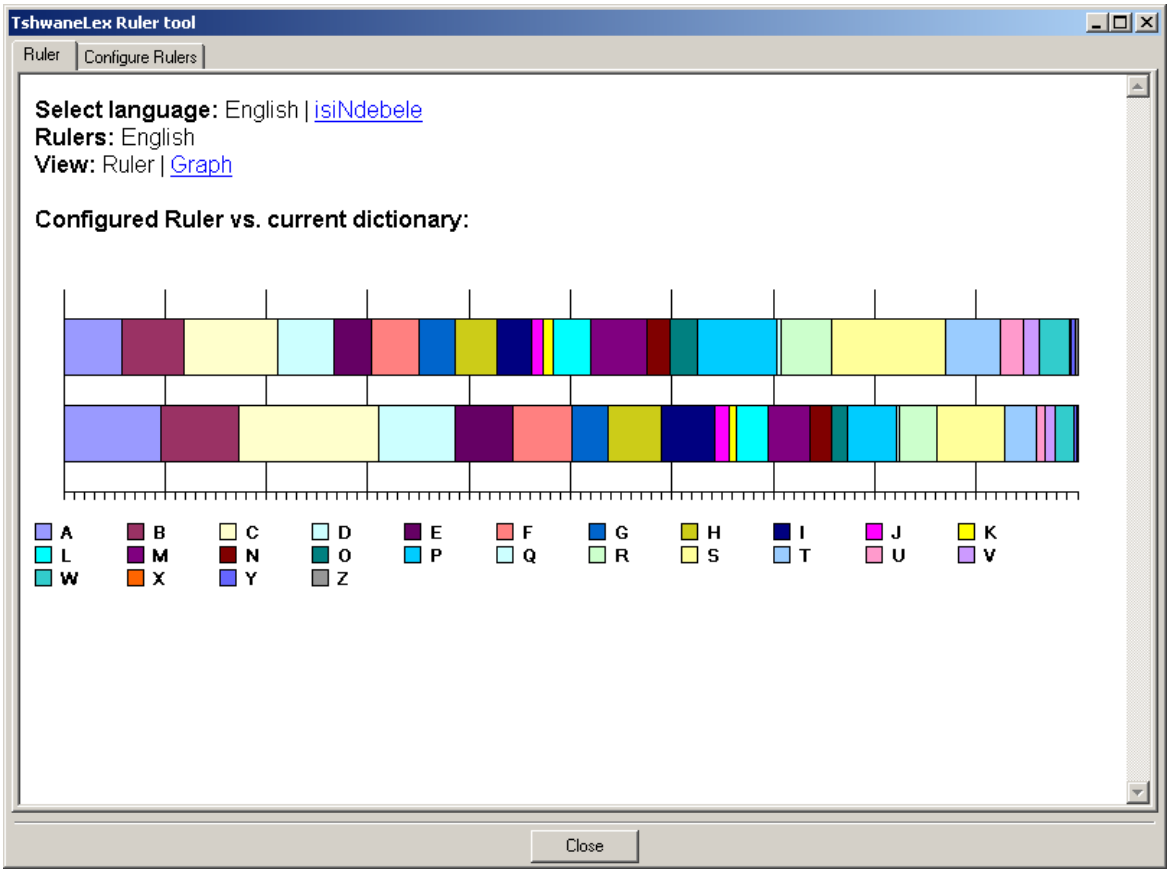


Figure 26: *TLex Ruler tool dialog, Ruler tab, "Ruler View"*
here comparing the general-language English Ruler (top) with actual English-side dictionary data (bottom)

The second (bottom) Ruler in Figure 26 shows the actual data of an English – isiNdebele – English dictionary that is being compiled. When one compares these two coloured strips, one immediately notices that there are quite some discrepancies indeed, with the first few categories of the dictionary being compiled far too large, and the second half under-treated. The Ruler visually pinpoints the problem areas (i.e. the problematic alphabetical categories), and one can proceed to redress the imbalance.

In TLex it is also possible to visually compare, not only the space allocation of the various alphabetical categories with the Ruler, but also to bring in the corresponding number of articles treated per alphabetical category. This can be seen from the screenshot below:

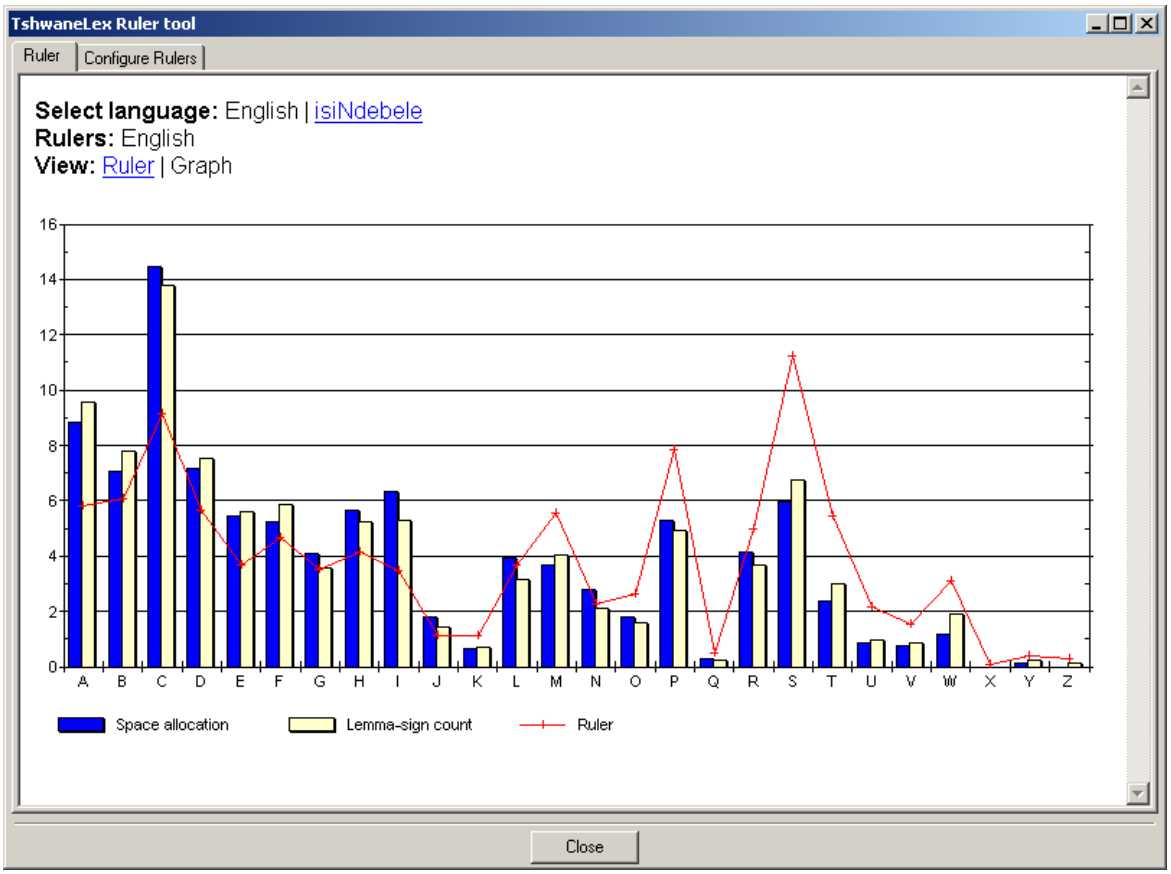


Figure 27: *TLex Ruler tool dialog, Ruler tab, "Graph View"*
comparing space allocation (blue bars), number of lemma signs (yellow bars) and the Ruler (red line)

From Figure 27, which displays the data for the same English – isiNdebele – English dictionary, the various over- and under-treatments are even clearer.

Given the heights of the blue and the yellow bars are always comparable for each alphabetical category, the *average article length* is constant in this dictionary. It is thus not the case that articles are becoming longer and longer in this dictionary, and that as a result fewer and fewer lemma signs are being treated. Rather, there are, in order to reflect the distribution of the English lexicon more correctly, simply too many articles under for example **C**, and too few under for example **S**.

Even if one does not have a Ruler for a particular language (the red line in Figure 27), one can still use the Ruler tool to study average article length in a dictionary, which, very often, will reveal that articles tend to grow in size as one proceeds through the alphabet.

Note that Rulers can be configured and used for the two sides of a bilingual dictionary, and also that data can first be filtered before using any Ruler.

Customising the Dictionary Grammar or Termbase Fields using the DTD [Advanced]

The preceding chapters presented a "gentle" overview of the use of the default features available in TLex/tlTerm. Starting with the current chapter, more advanced features are discussed. The first of these describes the fully customisable and built-in DTD (Document Type Definition) editor. This powerful tool, which is based on XML (eXtensible Markup Language) standards, allows you to tailor the dictionary grammar or termbase fields of any project. (Note that this is something many users would feel more comfortable allowing their 'IT experts', where available, to do. However, TLex/tlTerm allow you to do this yourself.)

The DTD editor dialog can be accessed from the "Dictionary/Customise DTD (dictionary grammar)" menu option.

Note: It is recommended that you make a **backup** of the database ("File/Create a backup") before making changes to the DTD.

Basics of Hierarchical Data Modelling

The basic structure of a dictionary article is hierarchical (as well as for a terminology entry, however, in the case of tlTerm, the hierarchies are usually far simpler). A lemma may consist of several word senses, and each of those word senses may in turn contain subsenses. Any of the senses or subsenses may contain usage examples or multi-word units ("subentries" in TLex), which themselves may again in turn contain one or more word senses. When creating an article, the lexicographer's task can be seen as consisting of two essentially separate but closely related sub-tasks: (1) To specify the basic skeletal *structure* of the article, such as the layout of word senses and usage examples, and (2) to flesh out that basic structure with *content*, such as the word sense definitions and usage example texts. One can apply a "tree" metaphor here: the structure may be seen as being like the "branches" of a tree, while the content itself are the "leaves" on the branches.

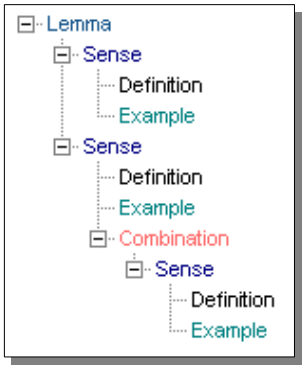


Figure 28: *Empty basic lemma tree structure ("tree without leaves" – elements only)*

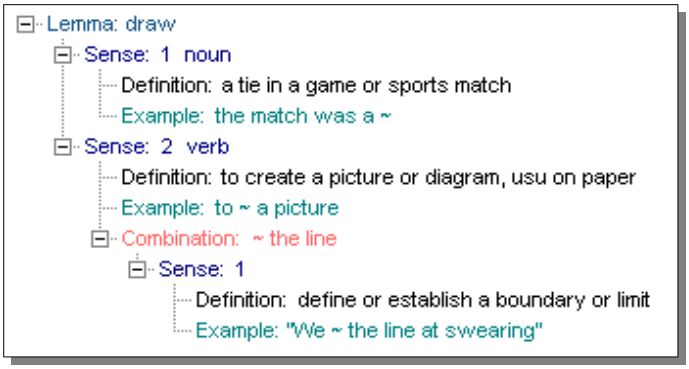


Figure 29: *Basic lemma tree structure with content filled in ("tree with leaves" – elements and attributes)*

Elements and Attributes

In TLex/tlTerm, the "root" and "branches" of the tree structure are called ***elements***. The text *content* values, which provide the actual *data*, are called ***attributes***. An element is essentially a *container* that stores, firstly, other child elements, and secondly, attributes. Different types of elements may be

defined, and for each element type there is a particular set of attributes associated with that element. For example, for the “Lemma” element, one may fill in attributes such as the lemma sign itself, the pronunciation field, and an etymology field. For a usage example (“Example”) element, there may be three attributes: the usage example itself, a translation (if a bilingual dictionary), and optionally the source (citation). For a “Definition” *element*, one may fill in the definition *attribute*.

The tree structure is edited using the Tree View control (see the section on Structuring Articles with the Tree View Control, in the chapter Getting Started with TLex: QuickStart Guide). To add a “Sense” to a lemma, for example, one right-clicks on the “Lemma” node in the Tree View and selects “Add: Sense”. The “leaves” of the tree, i.e. the contents, are filled in under the “Attributes (F1)” and “Attributes (F2)” tool windows. Thus one uses the Tree View primarily to edit the *structure* of an article, and the “Attributes” windows primarily to edit the *content* of element attributes.

When one clicks on any element in the Tree View, the “Attributes (F1)” window shows the attributes associated with the type of element one has clicked on, and one may immediately proceed with entering or modifying the relevant contents. For example, if you click on a usage example, “Attributes (F1)” shows the (three) fields that can be filled in for usage examples.

These links between (1) internal article structure, (2) Tree View, and (3) boxes / fields to be filled in by the user, constitute one of the crucial design features to ensure smooth and sound compilation. Only the relevant attributes are seen at any given point, so that the screen does not get clogged, and so that the potential for errors is minimised as only attributes allowed by the DTD are editable.

What is a DTD?

A DTD (Document Type Definition), also (loosely) known as a “schema”, is a description of the structure of the articles of a particular dictionary. Essentially, this is a description of the specific types of *elements* that a dictionary has, as well as the *attributes* that each of those elements may have. Every dictionary document needs a DTD, and this is typically mostly set up as completely as possible at the start of a dictionary project, with perhaps minor modifications later on. The DTD is then used while creating the dictionary, and by enforcing conformance to a well-designed DTD, one can ensure that the final dictionary follows a logical and consistent structure throughout.

When a new dictionary is created in TLex (i.e. using “File/New dictionary”), TLex creates a basic default DTD (likewise, for tlTerm). This may then be customised further by the user. Note, however, that a few DTD element and attribute types are fundamental to TLex and cannot be removed, although all elements and attributes may be renamed.

A DTD defines three basic things:

1. **Element types:** The types of elements that may appear in the dictionary.

2. **Element attributes:** The attributes associated with each element type.

3. **Element child relations:** For each element type, one may specify which other element types may be attached as children, for example a “Lemma” may have “Sense” elements attached to it.

Element Types

The DTD contains a *list of all the element types* that may appear in the dictionary.

In the DTD editor dialog, the full list of element types appears on the *left*, under the heading "Element types". See the screenshot below:

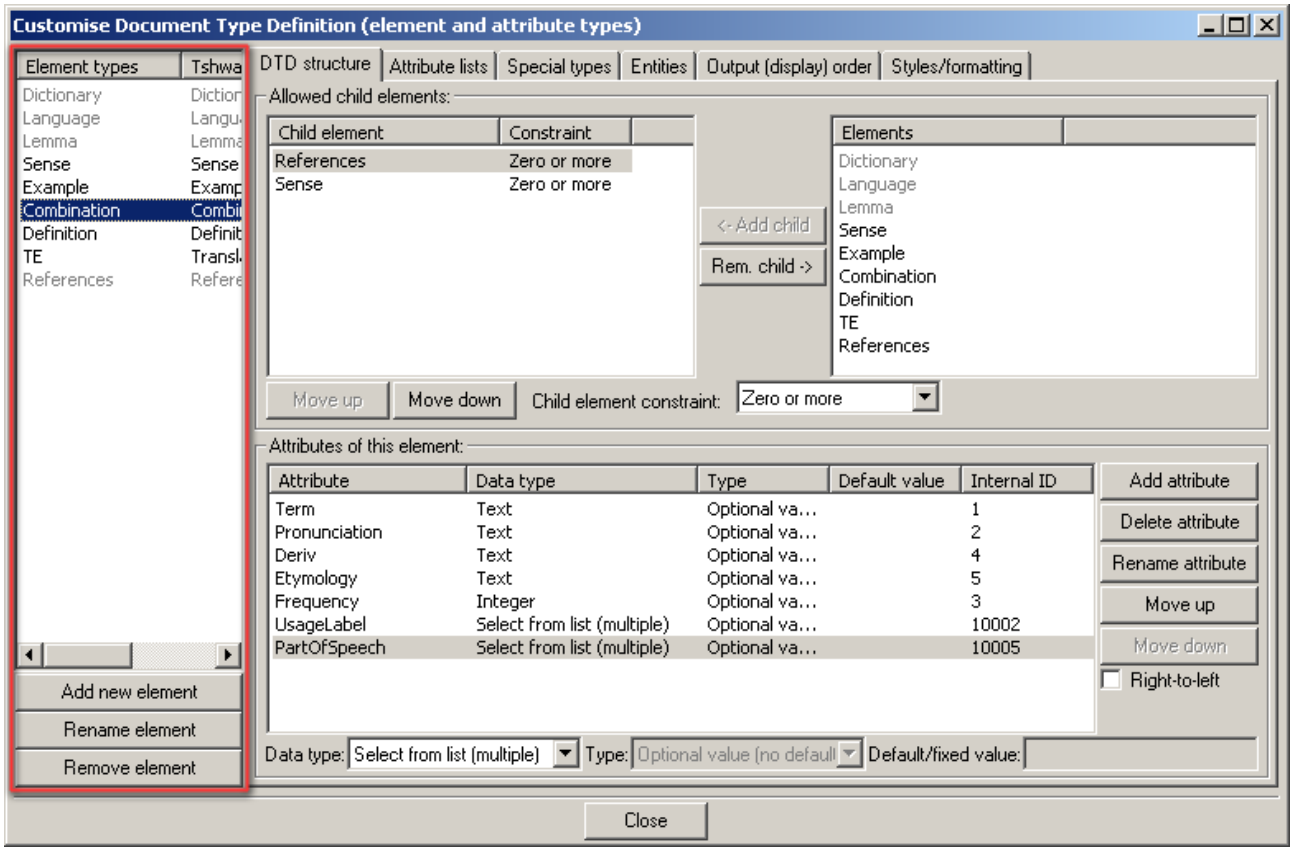


Figure 30: *DTD editor dialog, Element editing window: Editing the "Combination" element*

To create a new element type, click on the "Add new element" button below the element list, near the bottom left of the dialog.

Existing element types may be renamed or removed using the "Rename element" and "Remove element" buttons below the element list.

WARNING: If one removes an element type in the **DTD editor dialog**, *all existing elements of that type throughout the dictionary* will be removed.

Element Attributes

Each *element* in the DTD has its *own set of attributes*.

When an element is selected in the list of elements on the left of the DTD editor dialog, the "Attributes of this element" window on the "DTD structure" tab, shown in the screenshot below, displays a list of the attributes for that element. For example, if "Lemma" is selected, all lemma attributes are shown, such as "LemmaSign", "Pronunciation", "Etymology", etc. (These are the attributes that are shown under "Attributes (F1)" when a node of this element type is selected in the Tree View while editing the dictionary.)

New attributes may be added using the "Add attribute" button to the right of the list of attributes. Existing attributes may be renamed using the "Rename attribute" button. The order that attributes appear under "Attributes (F1)" may also be modified by using the "Move up" and "Move down"

buttons. (Note: This only affects the order that attributes appear under “Attributes (F1)”, and *not the output order* of attributes in the Preview Area or RTF, HTML, etc. exporters. The *output order* is configured separately, and will be explained later.)

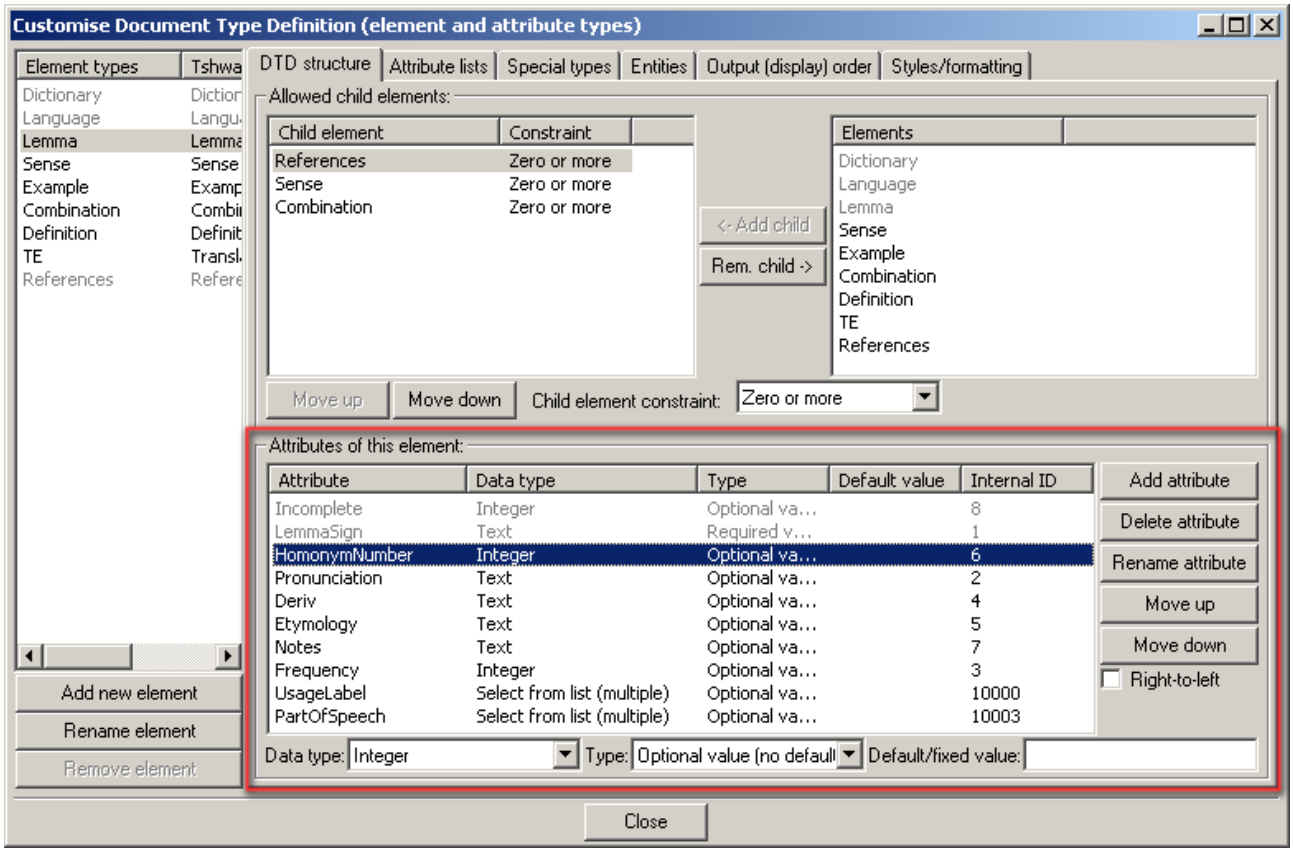


Figure 31: *“DTD structure” tab, Attribute editing window: Editing the attributes of the “Lemma” element*

Element Child Relations

The list of elements at the left of the DTD editor dialog is a “flat” list, i.e. it only describes *what* element types appear in the dictionary, but does not describe which element types are allowed to be attached to which other element types when building the hierarchical structure of each article in the Tree View using the right-click “Add:” commands. Clearly, only certain elements may be added to others, for example, it will make little sense to add a “Lemma” element beneath a usage example, so this is blocked. Thus, one needs to define element “child relations”, which describe the “allowed child element types” of each element. This is done in the “Allowed child elements” window of the “DTD structure” tab in the DTD editor dialog, shown in the screenshot below. This window contains two lists: the left one, with heading “Child element”, is a list of the currently allowed child elements for the selected element type. The list on the right is simply a list of *all* element types, from which one may choose to add allowed child elements.

For example, if one wishes to specify that a “Sense” element may have a “TE” (translation equivalent) element attached to it, then one would do the following:

1. Select “Sense” from the list on the very left of the DTD editor dialog.
2. Select “TE” from the list of all elements at the right of the “Allowed child elements” window.
3. Click on the “<- Add child” button.

The “TE” element will be added to the list of allowed child elements for “Sense”. Subsequently there will be an “Add: TE” command in the menu that appears when right-clicking on “Sense” in the Tree View.

One can remove an element from the list of allowed child elements for the selected element by selecting them in the “Child element” list and clicking the “Rem. child ->” button.

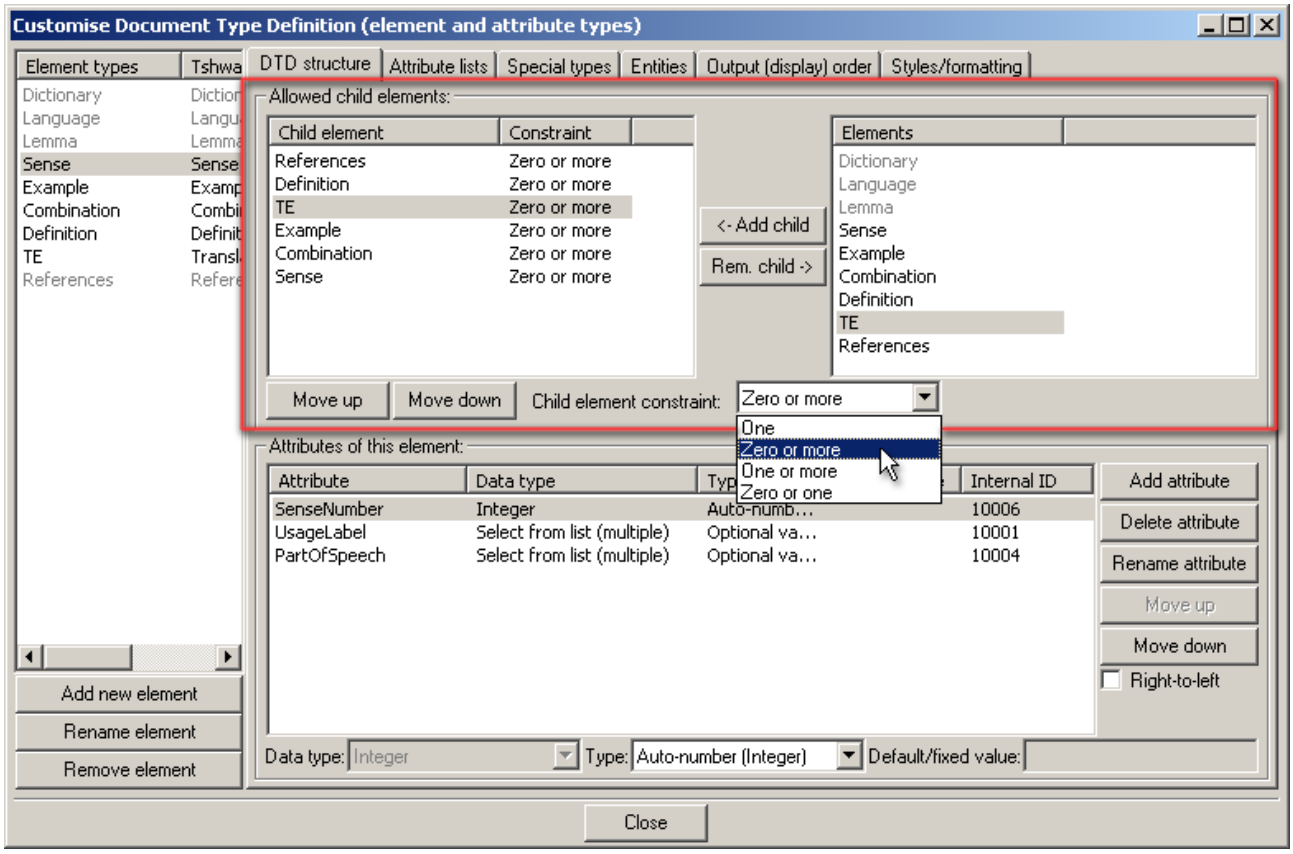


Figure 32: *“DTD structure” tab, Child elements editing window: Editing the list of allowed child elements of the “Sense” element*

Element Child Relation Constraints

Each “parent->child” element relation may also have a constraint associated with it, such as “this element may have *only one* child of this type”. For example, if the “Subentry->Sense” child relation specified that no more than one “Sense” may be attached to a “Subentry”, then TLex will not allow more than one “Sense” to be added to the “Subentry” element when editing in the Tree View, and this option will be “greyed out” in the right-click menu of the Tree View.

There are four possible constraint types, which may be selected using the “Child element constraint” drop-down list in the “Allowed child elements” window of the “DTD structure” tab:

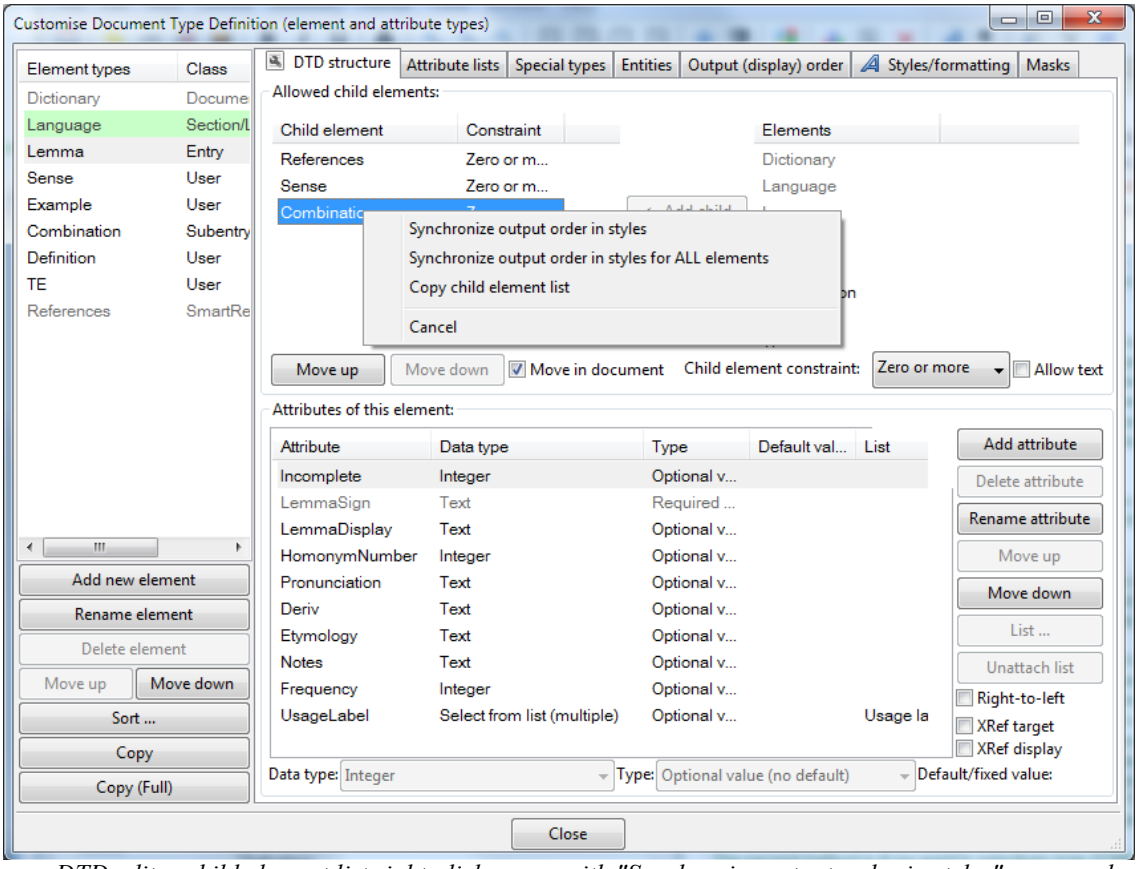
1. **One:** One child of this type only.
2. **Zero or more:** Zero or more children of this type.
3. **One or more:** One or more children of this type (“at least one”).
4. **Zero or one:** Either zero or one child of this type, but no more than one.

The order that child elements appear in the Tree View may also be modified by using the “Move up” and “Move down” buttons. (Note: This only affects the order that elements appear in the Tree View, and *not the output order* of child elements in the Preview Area or RTF, HTML, etc. exporters. The *output order* is configured separately, and will be explained later.)

Synchronizing the Child Element Order with the Output Order in the Styles

As mentioned above, the *output order* of child elements can be configured independently of the order the child elements are actually attached in the *data.* (The data order is typically based on the order the elements are specified in the DTD in the child elements list. One exception to this is when "PCDATA" is enabled for the parent element, then the order of child elements is based on the order they're specified by the user in the *Tree View* - this is a limitation of XML DTD's.)

The advantage of specifying *output orders* independently of the DTD order for child elements, is that it is more flexible and powerful, as it allows you to intermingle *attributes* of the element with *child elements* in the Preview / output generating system, e.g. "Attr1, ChildElem1, Attr2, ChildElem2". However, the disadvantage is that this creates a bit more maintenance work when configuring the DTD, as when you change the DTD order of child elements, you have to manually also change the *output order* specification in the Styles. In this case, the "**Synchronize output order in styles**" commands can be used to mitigate this, by automatically synchronizing the *output order* with the DTD order of child elements. These commands can be activated by right-clicking on the list of allowed child elements in the DTD editor.



33. Figure: DTD editor child element list right-click menu, with "Synchronize output order in styles" commands

These commands will try set the *output order* of child elements to be the same as their ordering in the 'allowed child elements' list. The first command does it for the selected element only, while the second does it for *all* elements in the DTD.

Note this won't necessarily work properly for cases where there are 'out of order' children, i.e. multiple instances of the same allowed child element (eg. "A,B,A").

It's also important to remain aware here that in the 'output orders', unlike in the actual DTD, the elements may be intermingled with attributes (due to the output order system that allows transforming the order on generating the output), e.g. "Attr1,ChildElem1,Attr2,ChildElem2" - if so, this function may leave some undesired orderings of child elements relative to attributes; these can only be resolved manually as this function cannot automatically know the correct desired ordering. The function tries to leave attribute ordering intact, but this isn't always possible.

If unsure, save a backup of your file before proceeding, and then verify that changes made were useful.

Note that if you have multiple *sets of styles*, these functions operate on the current set of styles only.

Copy Child Element List to Clipboard

The right-click menu described in the previous section also includes a command that copies the list of allowed child elements for the selected element to the system clipboard.

Special Element Types (TLex-internal)

Some element types have special meaning in TLex, in order to provide more advanced functionality, such as “smart cross-references”. The use of these special elements is not strictly required, but certain useful features of TLex will not be available if they are not used.

Dictionary and Language Elements

The “Dictionary” and “Language” element types appear in the list of elements but may be safely ignored; these are merely there for the broader structure of a TLex document.

Lemma Element

The “Lemma” element is the top-level element of each dictionary article. Its attribute “LemmaSign” cannot be removed, as its contents are used for automated sorting purposes.

References Element

The “References” element is a special element used to create TLex “smart cross-references”.

For more on the use of these, see the chapter on Editing Cross-references. The smart cross-reference types themselves can also be customised, see the chapter on Customising the Cross-reference Types for this aspect.

The TLex Default DTD

When you first create a new dictionary in TLex, a simple, sensible default "generic dictionary" DTD is created, in order to allow new users to get up and running quickly.

Elements

Following is a basic description of all elements included in the default DTD.

Note that the described intended usage is merely a guideline – you may adapt these elements for your own purposes as necessary.

Element	Description
Dictionary	Top-level, internal-use element for storing the entire dictionary document. Contains either one or two "Language" elements, for monolingual or bilingual databases respectively.
Language	Internal-use element for storing all lemmas of a monolingual dictionary, or for each side of a (semi-)bilingual dictionary. Contains only "Lemma" elements. [**Semi-bilingual dictionaries** are a hybrid type whereby dictionary articles are conceived to contain both translation equivalents and definitions right from the start.]
Lemma	Internal-use, top-level element of a single dictionary article.
Sense	A single word sense of a lemma or multi-word unit (when added as a child of a "Lemma" or "Subentry"), or a subsense (when added as a child of another "Sense").
Example	A usage example. Usage examples may be attached by default only to particular word senses, i.e. to "Sense" elements.
Subentry	(Previously called "Combination") - Used typically for multi-word units, e.g. idioms (see "draw the line" in the screenshots at the beginning of this chapter). This may by default be attached to the top-level "Lemma" element of an article, or to particular word senses (i.e. "Sense" elements).
Definition	This is typically used to store a single monolingual (same language) definition of a word or word sense. Zero or more definitions may be added to "Sense" elements. (Note that this is just a container for the Definition::Definition attribute, described below.)
TE	"TE" stands for "Translation Equivalent". This is typically used to store translation equivalents of a word in another language (in e.g. bilingualised dictionaries), or in the target language (in bilingual dictionaries). Zero or more translation equivalents may be added to "Sense" elements. (Note that this is just a container for the TE::TE attribute, described below.) [**Bilingualised dictionaries** are basically (existing) monolingual dictionaries to which translation equivalents are added on.]
References	This is a special element in TLex that can be used to create cross-references using the cross-reference editor dialog. By default, this may be attached to "Lemma", "Sense" or "Combination" elements.

Attributes

Following is a basic description of all attributes included in the default DTD. Note that the described intended usage is merely a guideline – you may adapt these attributes for your own purposes as necessary.

Attribute	Description
Lemma::Incomplete	Lemma "incomplete" flag (i.e. for the "Incomplete" checkbox under "Attributes (F1)").
Lemma::LemmaSign	The lemma sign (i.e. headword) of an article.
Lemma::HomonymNumber	The lemma homonym number. This is automatically calculated and filled in by TLex when multiple articles share the same orthographic form of the lemma sign (i.e. when headwords have the same spelling).
Lemma::Pronunciation	Pronunciation of the lemma sign.
Lemma::Deriv	Derivational information, e.g. "=can+not" for "can't".
Lemma::Etymology	Historical origin of the word, e.g. for the English word "safari" this might be "Swahili *safari* from Arabic *safar* journey".
Lemma::Notes	Internal *lexicographer's* notes, or notes from editor to lexicographer, e.g. "this article needs a better usage example" or "please check – the second sense looks incorrect".
Lemma::Frequency	The usage frequency of the lemma as calculated from a corpus.
Lemma::UsageLabel	Word usage information label, e.g. "formal", "colloquial", ...; "sports", "physics", ...
Lemma::PartOfSpeech	Word part of speech label, e.g. "noun".
Sense::SenseNumber	Sense number. This is *automatically calculated* by TLex.
Sense::UsageLabel	Sense usage information label, e.g. "formal", "colloquial", ...; "sports", "physics", ...
Sense::PartOfSpeech	Sense part of speech label, e.g. "noun".
Example::Example	Usage example. You can use the "~" (tilde character) as a substitute for the lemma sign of the article in the usage example, e.g. "We ~ the line at swearing" (in the article of "draw").
Example::Translation	Target language translation of usage example for bilingual dictionaries.
Example::Source	Source (citation) of usage example.
Subentry::LemmaSign	The multi-word unit itself, or lemma sign of a sublemma, e.g. "draw the line". You can use the "~" (tilde character) as a substitute for the lemma sign of the article, e.g. "~ the line" (in the article of "draw").
Subentry::Pronunciation	Pronunciation of the multi-word unit.
Subentry::Deriv	Derivational information of the multi-word unit.
Subentry::Etymology	Historical origin of the multi-word unit.
Subentry::Frequency	The usage frequency of the multi-word unit in a corpus.
Subentry::UsageLabel	Multi-word unit usage information label, e.g. "formal", "colloquial", ...; "sports", "physics", ...
Subentry::PartOfSpeech	Multi-word unit part of speech label, e.g. "noun".
Definition::Definition	This is typically used to store a monolingual (same language) definition of a word or word sense.

Attribute	Description
TE::TE	"TE" stands for "Translation Equivalent". This is typically used to store a *single* translation equivalent of a word in another language or in the target language. [Note that the idea is thus *not* to type several equivalents separated by, say, a comma in this field. For **synonyms** one will rather attach a new TE attribute to a single "TE" element (thus "TE::TE"), or fill in the "°TE (NEW)" field under "Attributes (F1)". For different, **polysemous senses** one will typically create a new sense first. The punctuation between synonyms and polysemous senses (say, commas vs. semi-colons) is decided on in the Styles System. See the chapter on the Styles System below for more information.]

Attribute Lists

Attribute lists are closed lists of pre-prepared attributes, from which one can simply select those attributes that need to be attached to certain elements. Basically, there are two different list types. In the first the user can only select one item from the list ("one of"), in the second zero or more items may be selected. For the second type, a difference is also made between "sorted" and "unsorted". For the sorted type, the order of the output of selected list items will always be the same as the order of the items defined centrally for the list. For the unsorted type, the order of the output of selected list items will be the same as the order in which they are selected by the user. Also note that any field can be converted from a free text field to a closed list (and vice versa) at any time.

The "one of" attribute lists are found under "Format (F1)", see e.g. Figure 10, while the two types of "multiple" attribute lists are found under "Format (F2)", see e.g. Figure 11.

The second tab on the DTD editor dialog, the "Attribute lists" tab, is used for the management of the various lists in the dictionary database. This tab consists of four windows, as indicated on the screenshot below:

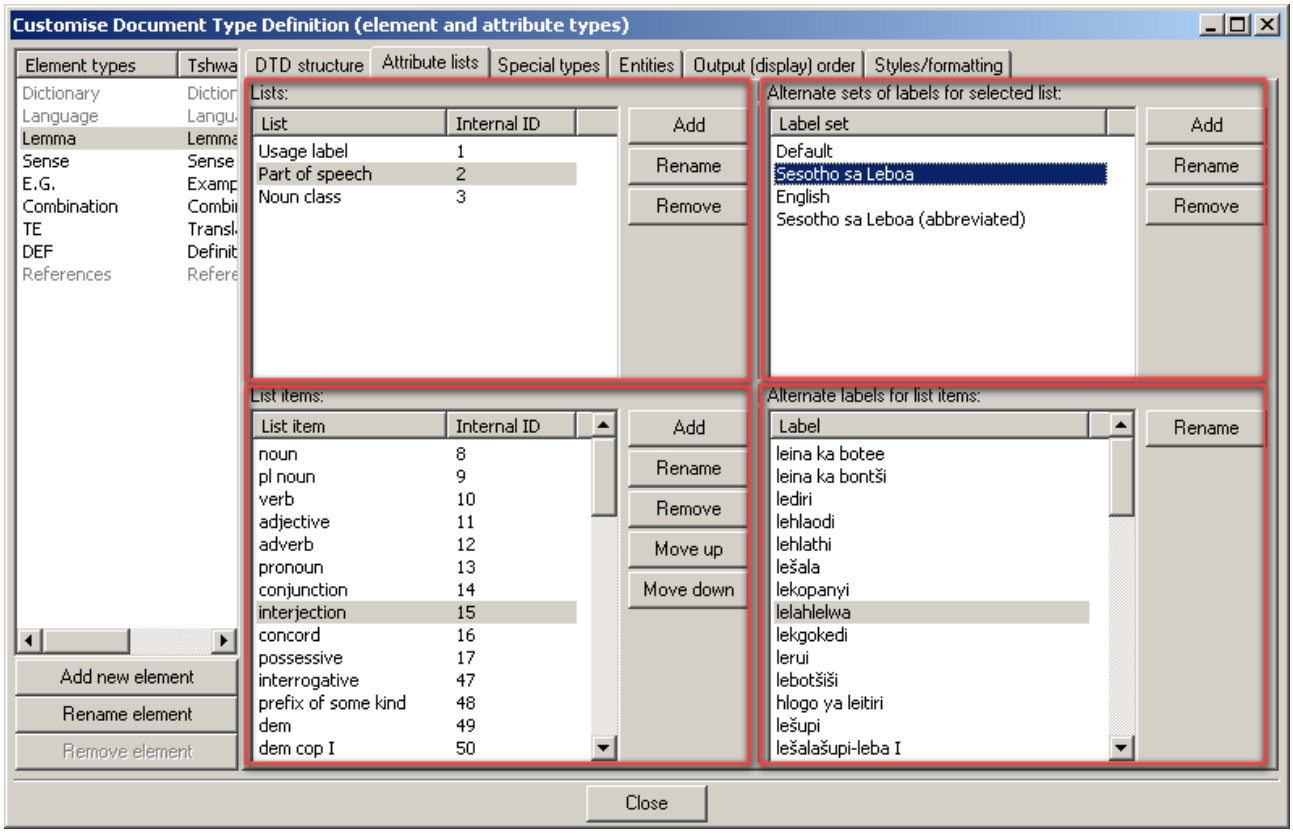


Figure 34: *"Attribute lists" tab in the DTD editor dialog*

The top-left window, called "Lists", simply enumerates the names of the various attribute lists. Existing attribute lists can be renamed or removed, and new ones added. The dictionary database in the screenshot contains three lists, namely "Usage label", "Part of speech" and "Noun class". When you stand on the name of a particular list in the "Lists" window, the list items for that particular attribute list are revealed immediately underneath, in the bottom-left window labelled "List items". Here too, existing items can be renamed or removed, and new ones added. The order of the list items can be changed with the "Move up" and "Move down" buttons. This order is also the order in which the items will appear under F1 and F2.

Once one has prepared the data under "Lists" and "List items", one basically has enough to start using the attribute lists. What remains to be done, however, is to link those attribute lists to the elements on which these list attributes are allowed to appear. This is done on the "DTD structure" tab. For example, if one wishes to attach a "Part of speech" list to the "Combination" element, then one will first need to create a new attribute for the "Combination" element, e.g. "PartOfSpeech", and, under "Data type", select one of the list types. Say one opts for the type "Select from list (multiple)", as was done for the dictionary shown in Figure Error: Reference source not found, then a new dialog is presented, from which one can select the appropriate attribute list, as shown below:

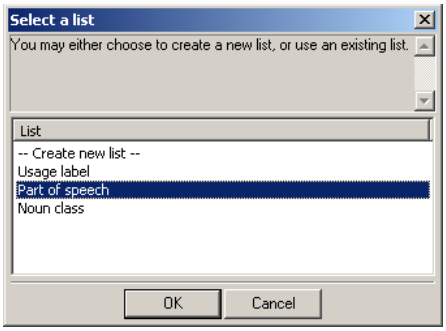


Figure 35: *Attaching an attribute list to an element in the DTD*

Choosing the attribute list “Part of speech” and clicking “OK” will attach that attribute list to the “Combination” element. Given a “multiple” type was chosen, this list will be available under F2. A single list can be reused as many times as one wants. So, the same “Part of speech” list can also be attached to for example the “Lemma” element and the “Sense” element. This is for instance the case in Figures 31 and 32 respectively.

The power of using attribute lists for dictionary compilation is obvious: they save time during the editing of the data (as items can simply be selected from drop-down menus (F1) and/or selected by ticking checkboxes (F2), and should thus never be typed in when compiling articles), they ensure consistency (e.g. ‘preposition’ will always be ‘prep.’, and not arbitrarily ‘prep’ or ‘prep.’ or ‘pp’ or ‘pp.’ or ‘preposition’, etc.), and spelling errors are avoided (as the lists only need to be prepared once). Moreover, all list attributes are available for the “Filter (F5)” tool, which means that one can easily extract all lemmas marked as, say, belonging to the sports domain.

However, in TLex/tlterm the use of (closed) attribute lists has been taken one step further. Firstly, they can be expanded at any time, by simply returning to the DTD editor dialog, and modifying the existing lists. Secondly, each of the lists can have as many (customisable) “linked variant / alternative lists” as one wants, which effectively means that one can *dynamically customise the language of the metalanguage* with just a few clicks. If one prepares both a long and a short form of the usage labels, then one can use the former for a desktop edition and the latter for a pocket edition of the same data. If, for bilingual lexicography, one has prepared the part-of-speech lists in the source as well as the target language, then versions with different metalanguages can be output depending on the intended market. In an electronic or online environment, dynamic metalanguage customisation can even be realised in real time, allowing for truly instantaneous tailoring.

Names for parallel lists are given in the top-right window of the “Attribute lists” tab, which is called “Alternate sets of labels for selected list”. The alternative items themselves are entered underneath, in the bottom-right window labelled “Alternate labels for list items”. In Figure 34, for example, the Sesotho sa Leboa (Northern Sotho) equivalents for the English part-of-speech items are shown.

Once an attribute list and its alternates have been set up in the DTD, they immediately become available for compilation, as may be seen from the F2 window in the screenshot below (where “leina ka botee” is the Northern Sotho equivalent of “(singular) noun”):

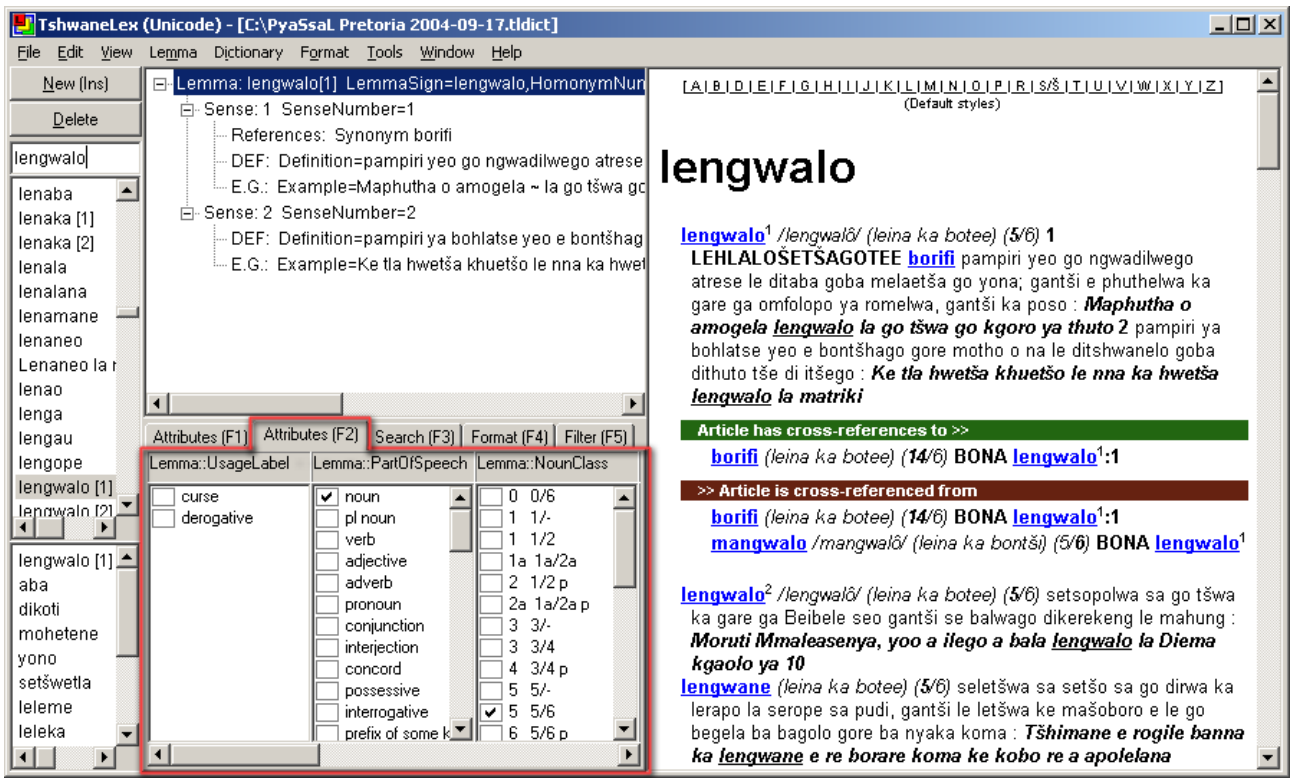


Figure 36: *Selecting list items from attribute lists under "Format (F2)" for an explanatory Northern Sotho dictionary [Data online at: http://africanlanguages.com/psl/]*

Swapping to an alternate label set for attribute lists may easily be done under the F4 window, as shown below:

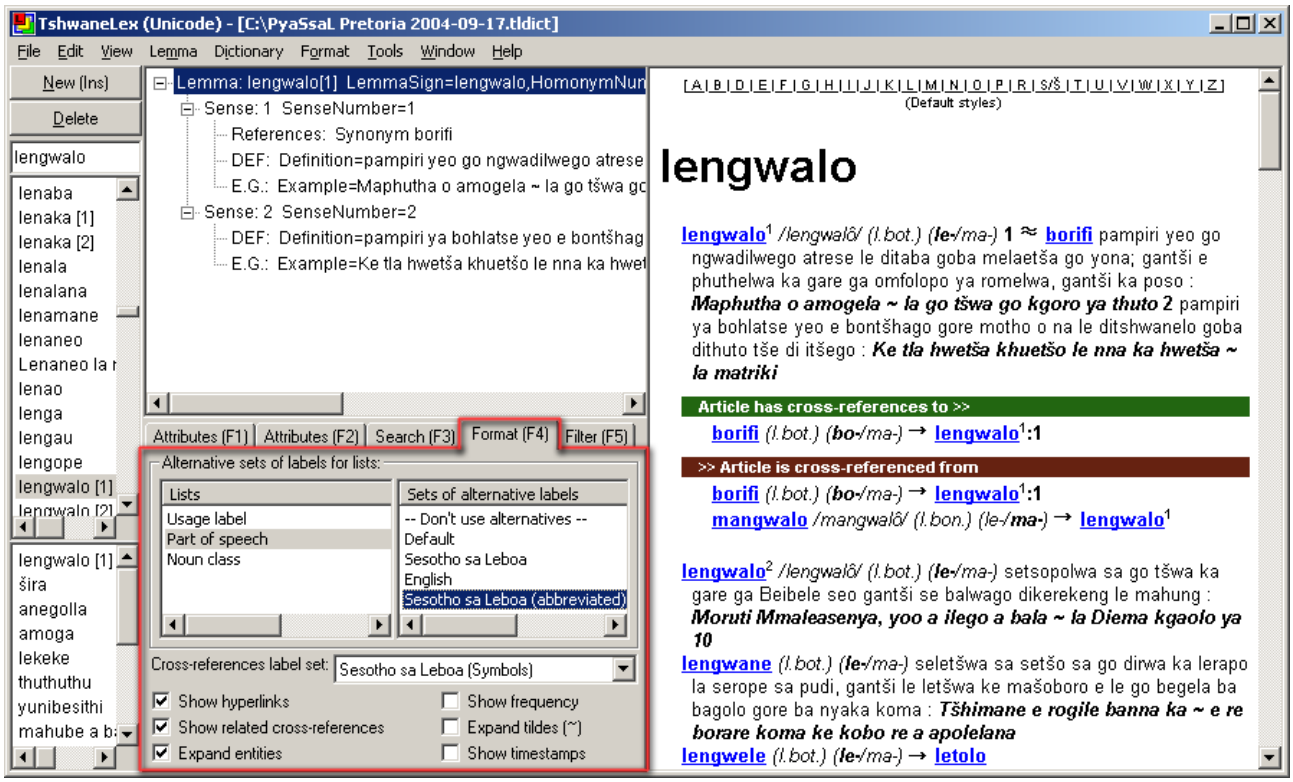


Figure 37: *Selecting alternate label sets for attribute lists under "Format(F4)" for an explanatory Northern Sotho dictionary [Data online at: http://africanlanguages.com/psl/]*

Changes take immediate effect throughout the entire dictionary database, as seen in the Preview Area. These settings are also used when exporting data.

Multimedia (Audio and Images)

Both sound recordings and images can be linked to any element in the DTD. To do so, it suffices to select the “Sound file (Text)” attribute type, respectively the “Image file (Text)” attribute type, on the “DTD structure” tab of the DTD editor dialog. For a screenshot of the relevant section of the “DTD structure” tab, see Figure 48.

All sound files and images are stored in a central place. This place is defined by filling in the “Sound/images path” input box of the “Dictionary/Properties” menu option. See Figure Error: Reference source not found for a screenshot of that dialog.

For an example of a database with both sound files (in this case “audio pronunciations”) and image files, see Figure 9. Sample 'Images' and 'Sounds' databases are also included in the 'Projects/Samples' folder.

Special Element and Attribute Types (User-configurable)

The advanced dictionary-compilation-specific functionality built into TLex – such as Linked View (whereby implicit links between the two sides of a bidirectional bilingual dictionary are automatically made visible for the user), Automatic Reversal (whereby single articles or even an entire (semi-)bilingual dictionary may be reversed by the software), or Cross-reference Tracking (whereby cross-reference integrity is ensured at all times by means of the automatic updating of target homonym and sense numbers whenever these change) – implies that the software makes certain assumptions on the use of a few DTD elements and attributes. Although fundamental to the proper functioning of TLex, changes can nonetheless be made to the default settings, using the “Special types” tab available from the DTD editor dialog.

As can be seen from the screenshot below, the elements and attributes used by TLex which can nonetheless be reassigned in the DTD concern the “Translation equivalent”, “Frequency”, “Sense”, “Display lemma sign” and “Internal 'notes'” fields.

Note: It is recommended that you make a **backup** of the dictionary database (“File/Create a backup”) before reconfiguring which elements and attributes have ‘special’ meaning in the DTD.

Translation Equivalent (TE)

The default element::attribute “TE::TE” is used by TLex when performing tasks such as automatic lemma reversal, as well as showing related lemmas on the other side of the dictionary when in Linked View. From an *n*-lingual database, *n* bidirectional bilingual dictionaries each could be produced (by exporting data to *n* databases).

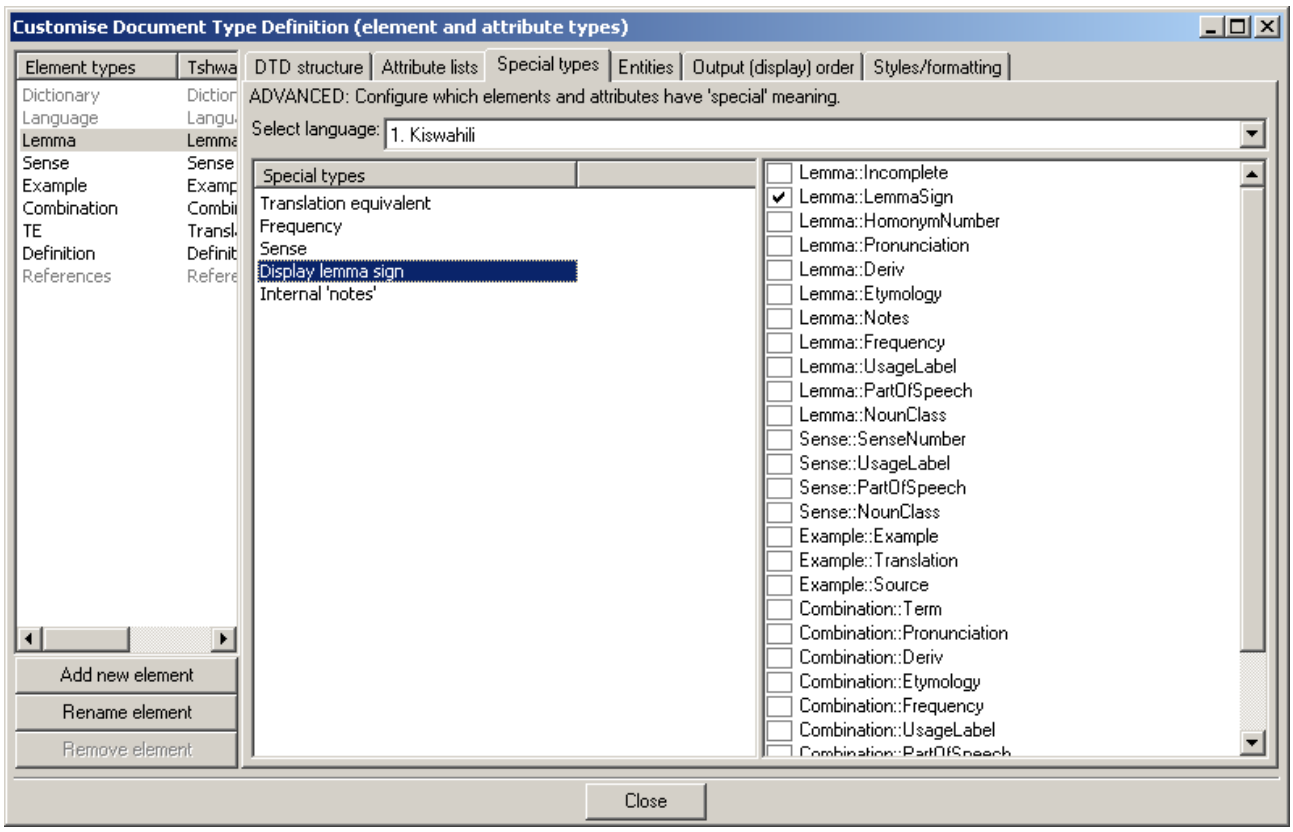


Figure 38: *"Special types" tab in the DTD editor dialog, for a bilingual Kiswahili – English dictionary [Data online at: http://africanlanguages.com/kdp/]*

Frequency

The default element::attribute "Lemma::Frequency" contains (corpus-derived) frequency data. One may instead wish to present frequency data for larger structures (e.g. combining homonym counts) or smaller ones (e.g. on sense level).

Sense

The default "Sense" element is a core component for each polysemous article. It typically contains an attribute for automatic numbering, and has a "TE" and/or "Definition" as child elements. The automated cross-reference tracking needs to know which elements are senses.

Display Lemma Sign

This setting allows you to configure which attributes in the DTD are, semantically, lemma signs (or at least 'display lemma signs'). This is used to generate the hyperlinks on the 'headwords' in the Preview Area. Thus if one adds a new attribute or element::attribute "DisplayLemmaSign" for, for instance, the full edition vs. a pocket edition, one can 'tell' TLex that "DisplayLemmaSign" should be a hyperlink in the Preview Area. These can be configured for each side of the dictionary separately (cf. the screenshot above).

Internal 'Notes'

The default element::attribute “Lemma::Notes” is used for internal lexicographer/editor’s notes. Instead of only attaching “Notes” to the head of an article, one may wish to, for instance, have the possibility to add notes at sense level too. TLex needs to know which fields are reserved for internal notes, as notes can for example be automatically suppressed when outputting data.

DTD Templates

It is possible to save the DTD of a particular dictionary project with the “Dictionary/Save DTD template” menu option. This DTD template can then be loaded into a new dictionary (“File/New dictionary”). See the “Load DTD template from file” section of Figure Error: Reference source not found in this regard.

Customising the Cross-reference Types

Renaming, Removing and Adding Cross-reference Types

When a new dictionary is created in TLex ("File/New dictionary"), the default cross-reference types are as shown in the screenshot below:

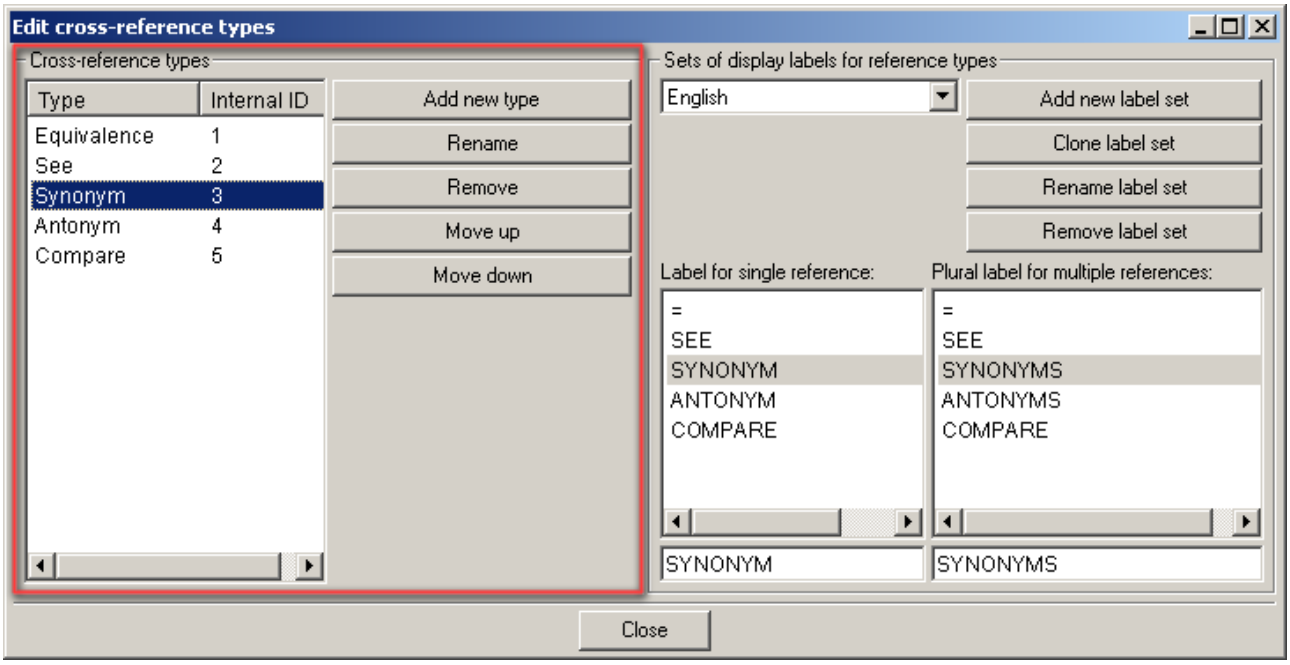


Figure 39: *Cross-reference types editor dialog, with default values*

This screenshot shows the cross-reference types editor, accessible from the "Dictionary/Edit cross-reference types" menu option. Existing cross-reference types can be renamed or removed. New cross-reference types can be added by clicking on the "Add new type" button.

Changing the Output Order of Cross-reference Types

When cross-references are displayed in the Preview Area or output, they are automatically grouped by cross-reference *type*. The order in which these groups are output corresponds to the order in which they appear in the list of cross-reference types in the cross-reference types editor. Thus, a single "References" element that contains both an "Equivalence"/"=" and a "See" cross-reference, will by default first display the "=" cross-reference, and then the "SEE" cross-reference. This order can be changed by using the "Move up" and "Move down" buttons in the cross-reference types editor (see screenshot above). You can thus configure the order in one place, and the change will immediately reflect throughout the entire dictionary.

Creating Singular and Plural Labels for Cross-reference Types

Also note that, for each reference marker, provision has been made for the creation of a singular as well as a plural form. See Figures 39 and 40 for examples. TLex will automatically use the plural

form when cross-references are made to several items of the same type. See in this regard for instance Figures 14 and 15.

Customising the Language used to Display Cross-reference Types

Analogous to the attribute lists, to which any number of parallel lists can be linked, the cross-reference types too may come in sets, and this for a multitude of useful purposes:

- Firstly, where one might find the cross-reference texts written out in full in a multi-volume edition of a certain dictionary, one will try to shorten those reference markers for a pocket edition, either by abbreviating the texts, or through the use of symbols. To do this, one already needs three alternate/variant lists.

- Secondly, in a bilingual dictionary one may wish to be able to configure the cross-reference type information (e.g. “see”, “compare”) to be displayed in a different language (e.g. “bona”, “bapetša” in Northern Sotho), depending on the dictionary’s target market. If this is done, it forms part of the framework to customise the language of the metalanguage.

- Thirdly, when publishing in electronic or online media, one may wish to *dynamically* change the language used to display the cross-reference type information, depending on the preference of the dictionary user. For example, a mother-tongue speaker of Northern Sotho may prefer to see the cross-reference type labels in Northern Sotho.

TLex allows you to create sets of alternate cross-reference type labels, in different formats and in different languages. In the screenshot below, three sets of display labels for cross-reference types have been prepared:

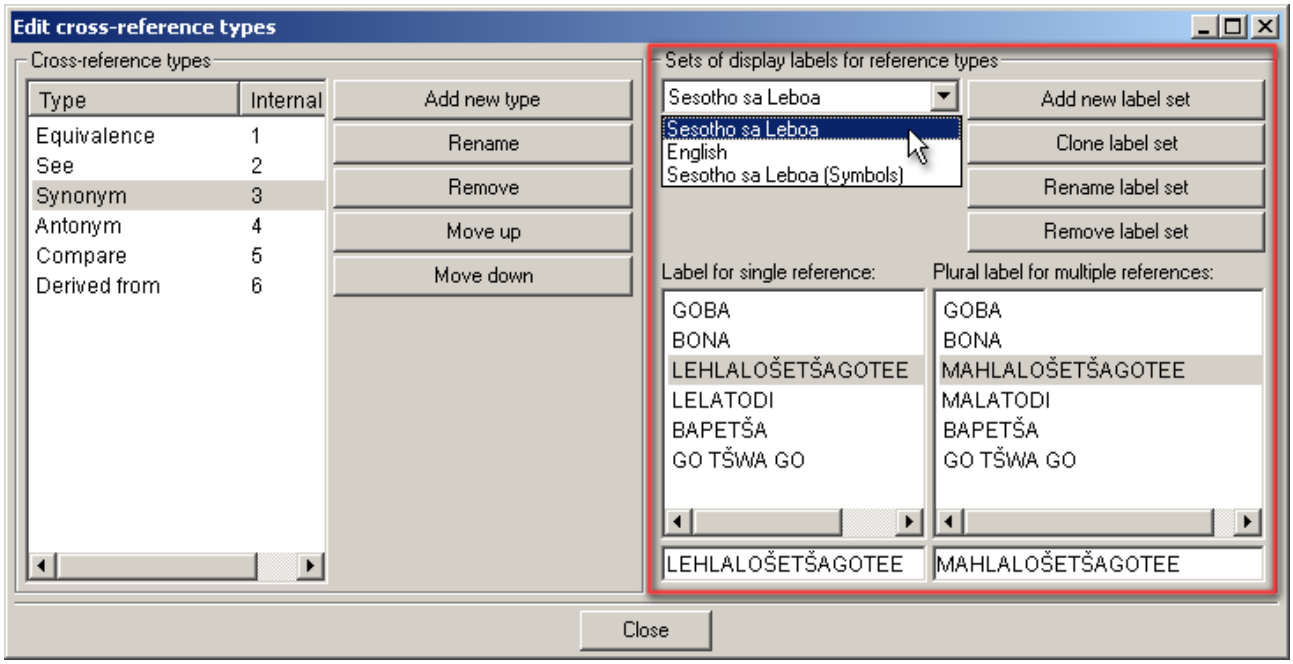


Figure 40: *Configuring alternate sets of display labels for cross-reference types, for an explanatory Northern Sotho dictionary [Data online at: http://africanlanguages.com/psl/]*

With the “Cross-references label set” setting under “Format (F4)”, illustrated below, one can then dynamically toggle between the sets of labels, and they will immediately change throughout the dictionary.

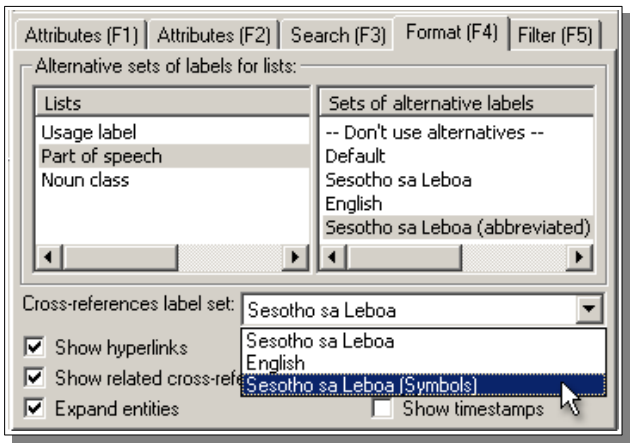


Figure 41: *Selecting/using an alternate set of display labels for cross-reference types, for an explanatory Northern Sotho dictionary [Data online at: http://africanlanguages.com/psl/]*

Customising the Alphabetic Sorting

Overview

TLex automatically sorts the articles in your dictionary, freeing you from having to do so. However, many different methods of sorting exist, and often many even for the same language. In order for TLex to support any possible sorting method that may be desired, an extendibility mechanism was included whereby new "sorting plug-ins" can be created. Thus, support for any sorting method (e.g. by radical/stroke count or by pinyin romanised form for Chinese) can be added to TLex.

Table-based Sorting

The default sorting method/"plug-in" supported by TLex is a configurable four-pass table-based sorting system (based on the ISO 14651 standard) that can be used for most Latin-based alphabets. The four passes are used for checking different characteristics that may take precedence over one another when two strings are compared to determine their order in the dictionary.

For example, in pass one, the strings are typically first compared using the so-called "base alphabet", that is, the underlying "alphabetic characters" of the letters, as if there were no diacritics.

Only if the underlying letters are the same, are the diacritics in the string then compared in a second pass. This allows all "related" characters to be sorted together, e.g. all "o" forms ("o", "ö", "ò", "ó", "ô", etc.), meaning, for example, that the French lemma signs "roder" and "rôder" will follow one another in the dictionary, as the dictionary user would likely expect. Thus all "o" forms are, in this case, *primarily* sorted as if they were just "o", and only *secondarily* sorted on the differences in diacritics.

If both the underlying alphabetic characters *and* the diacritics are the same, then a third pass checks for uppercase/lowercase differences (e.g. the French lemma signs "Moïse" vs. "moïse").

Finally, if there are no case differences, the fourth pass compares all other "non-alphabetic" characters, e.g. spaces, dashes and other punctuation symbols (these characters in the fourth pass are known as "ignorables").

Configuring Table-based Sorting

The specific alphabetic characters and diacritics that should be used during comparison, as well as the order of the letters, is naturally language-dependent. For example, in Estonian, "z" is sorted before "t". Additionally, for some languages, certain diacritic characters *should* be sorted as separate alphabetic characters even when the underlying form is the same as another character. Again for Estonian, "ä", "ö" and "ü" are separate letters that are sorted between "w" and "x", and not together with "a", "o" or "u" (see Figure 42 below). Thus, the entire system can be fully configured in "tables" – hence the name "table-based sorting".

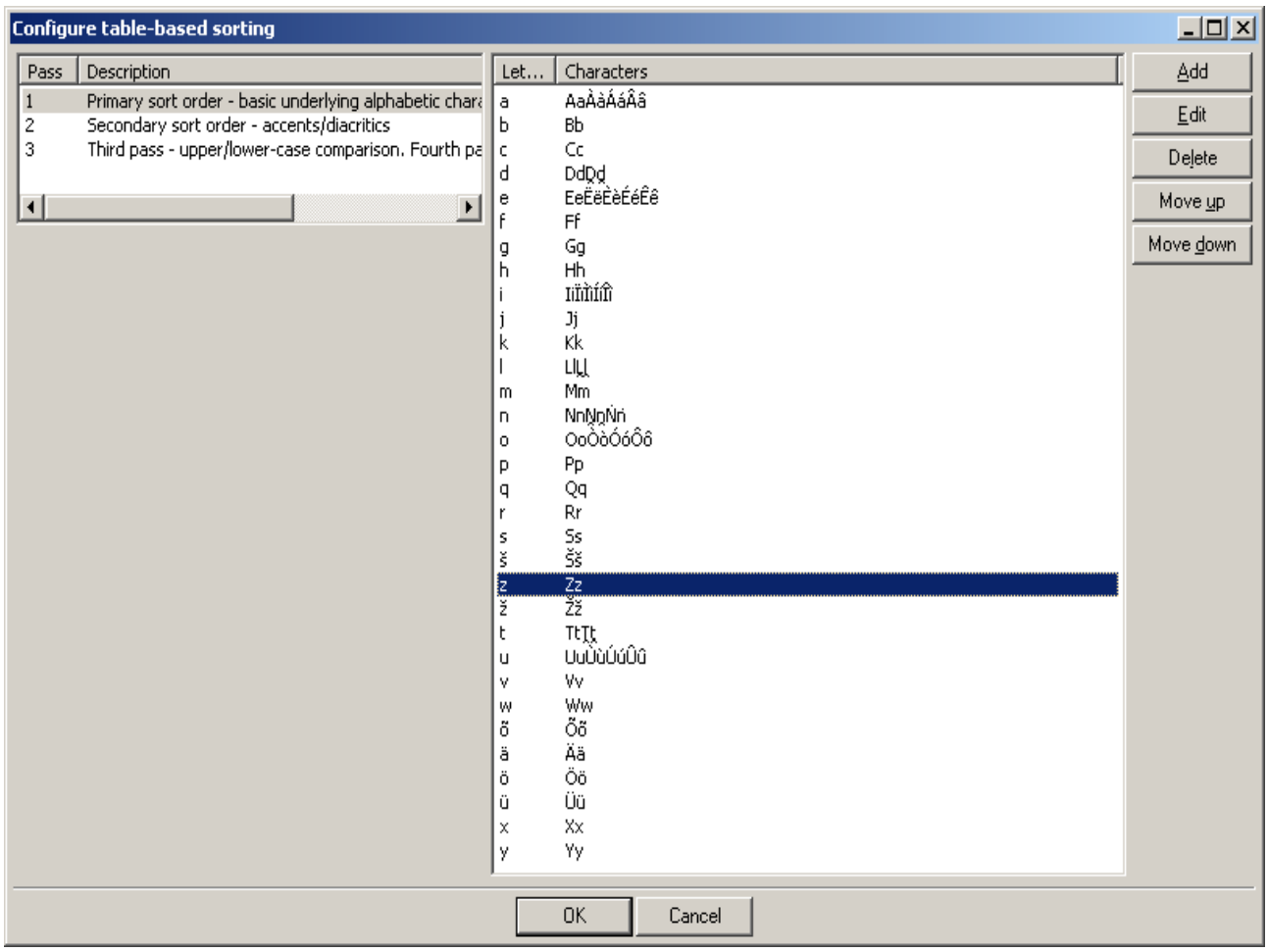


Figure 42: *First table/pass of sorting configuration for the Estonian alphabet*

In order to edit the tables that define the sorting behaviour for the table-based sorting system, select the menu option “Dictionary/Configure sorting”, then click on the table-based sorting method in the list under “Configured sorting methods”, and click on the “Configure ...” button. The configuration dialog for the table-based sorting will appear, allowing you to define alphabetic characters, their ordering, and the precedence of for instance diacritics or case differences.

Note: Table-based sorting is rather complex; a good way to familiarise yourself with the system is to study and depart from the existing default configuration.

Selecting Other Sort Plug-ins

The table-based sorting system is just one “sorting plug-in” in TLex. A different plug-in would be used to sort, for example, Chinese characters.

In order to make use of a different sort plug-in, or even to use a differently configured instance of the same table-based sorting plug-in, firstly open the “Configure sorting” dialog, shown in Figure 43.

The bottom half of the sorting configuration dialog displays a list of all sorting configurations which have been set up for this dictionary. The top half of the dialog displays which of these sorting configurations are actually *currently in use* by the language(s) in your dictionary. In the above screenshot, depicting an Estonian – Chinese bilingual dictionary, the Estonian side of the dictionary

has been configured to use the table-based sort plug-in, while the Chinese side of the dictionary has been configured to use the “Chinese - Radical / stroke count” plug-in.

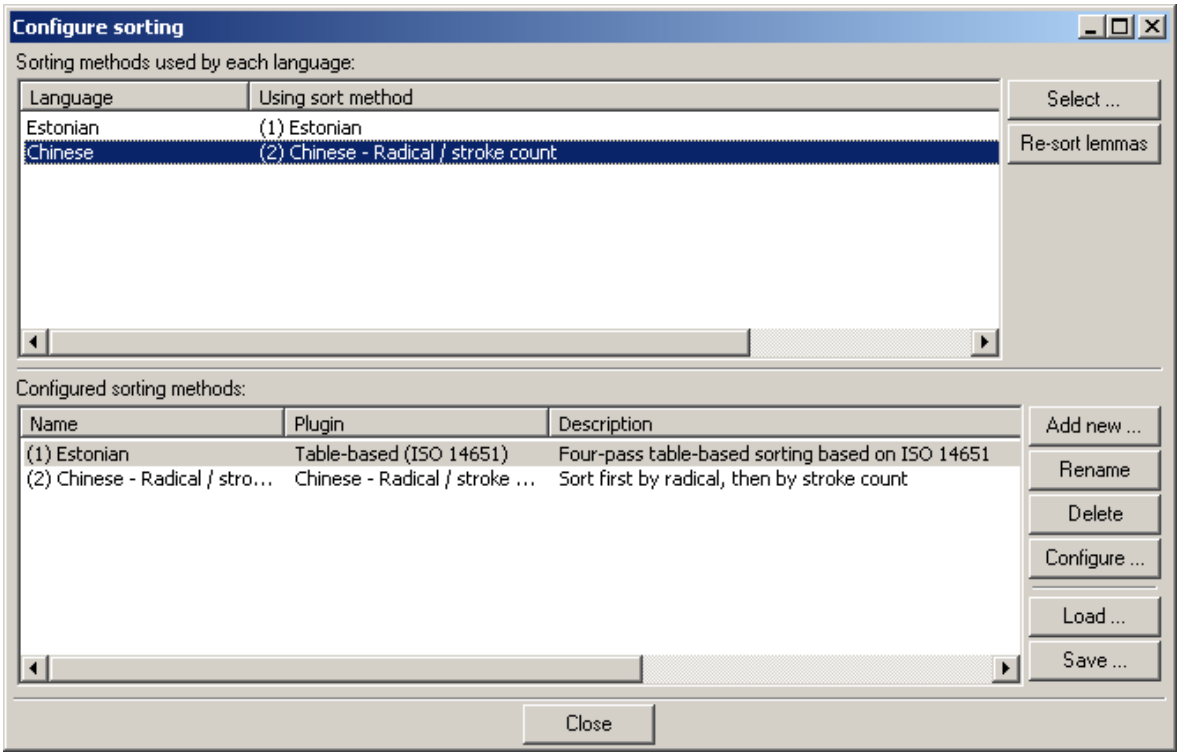


Figure 43: *Sorting configuration dialog*

The steps for adding and selecting the Chinese sorting method are as follows:

- Click on the “Add new ...” button.
- Select the desired plug-in to use from the list, namely “Chinese - Radical / stroke count”.
- Click “OK”.
- Optionally enter a name for this sorting configuration, e.g. “Chinese sort”.
- In the “Language” list in the top half of the dialog, click on the language you would like to have sorted with this newly selected plug-in, namely “Chinese” in this case.
- Click on the “Select ...” button to choose from the list of “configured sorting methods” in the bottom half of the dialog.
- Select the sorting configuration from the list that appears, here “Chinese sort”, and click “OK”.

The Chinese side of the dictionary will then automatically be resorted with the new sorting method.

Note that you *must* first configure a sorting method in the bottom half of the dialog *before* you can select it for use in the top half of the dialog.

Also note that, unlike the table-based sorting, the Chinese sorting plug-in has no further end-user configuration options; hence, the “Configure ...” button will be disabled for this sorting method.

Loading/Saving Sort Configurations

Once you have configured a sort method, you may want to re-use the configuration for other projects. The “Save ...” and “Load ...” buttons allow you to save the current configuration to a “sorting configuration file”, or load a previously-saved configuration.

Overview

The Styles System is used to "transform" the structured data of a dictionary article into the corresponding output in the Preview Area (and when exporting to formats such as RTF, HTML, etc.):

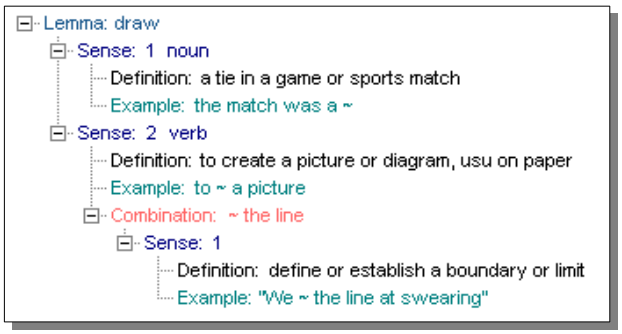


Figure 44: *Structured data of an article in the Tree View*

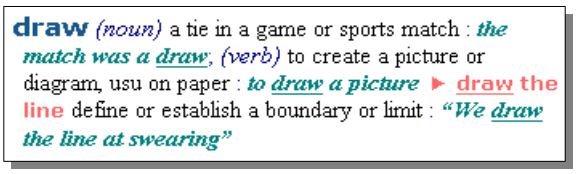


Figure 45: *Corresponding formatted article in the Article Preview*

Every element and every attribute in may have its own style. The styles in can be modified by opening the **Styles dialog**, accessible from the "Format/Styles..." menu option. The Styles dialog consists of three tabs: "Styles/formatting", "Output (display) order" and "Entities".

Styles/formatting

In addition to being available from the "Format/Styles..." menu option, the "Styles/formatting" tab in the Styles dialog can also be accessed by clicking the "Styles/formatting" tab at the top of the DTD editor dialog, and even by right-clicking on any element in the Tree View and choosing the "Modify styles/Edit style" menu option. The "Styles/formatting" tab (see the screenshot below) shows a list of all available styles on the left, under the header "Styles". Element styles are highlighted in grey, and the attribute styles for each element are shown beneath it. Selecting an element or an attribute from this list enables one to modify its style properties. Note that when this dialog is called up from the Tree View, the dialog will open with the styles/formatting of the element one has clicked highlighted.

Basic Formatting Options

A "style" allows various formatting options (such as font face, font size, bold, italics, underline, superscript, small caps, etc.) to be chosen for each element or attribute. The style can be changed in one place, and the change will immediately take effect throughout the entire dictionary for all occurrences of the corresponding element or attribute. These style changes are also applied immediately in the Preview Area, thus one may see the effects of style changes *while* one is changing the styles. (Resize and/or move the window of the Styles dialog to see (part of) the Preview Area.)

In this regard the Styles System thus functions in a similar way to the styles systems found in word processors such as OpenOffice.org and Microsoft Word. Further, when generating RTF output,

corresponding styles are created in the word processor document, allowing the styles to be further manipulated, if necessary, in those packages.

If a style property (e.g. bold) on an element or attribute is not specifically set, it 'inherits' the current value for that style property from its 'parent' element during the output process. Thus if one enables **bold** for the "Lemma" element, the entire article will become bold.

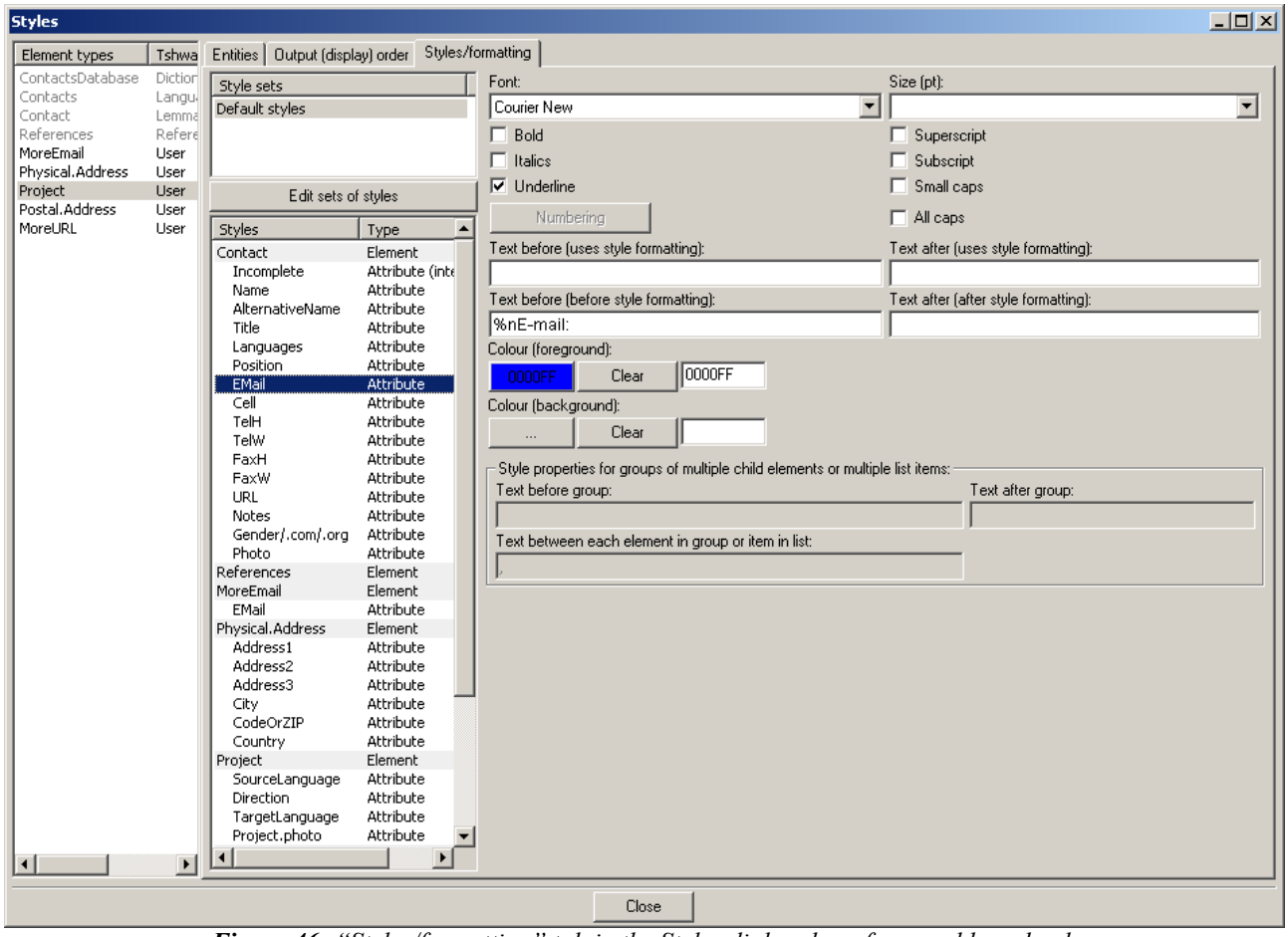


Figure 46: *"Styles/formatting" tab in the Styles dialog, here for an address book*

Font Names Starting With "@"

Some of the fonts in the drop-down list of fonts are prefixed with a "@" character. Generally, you *should not use these*; they are for top-to-bottom text for certain languages, and you will probably see spacing problems in the output if you use these. Use the version of a font name without the "@" character.

Colours

Each element and attribute can have its own foreground and background colours. Colours can be chosen from a user-friendly colour chart, which is shown below. In this case **blue** is selected as the foreground colour of the "EMail" attribute seen in Figure 46.

Once one has chosen a colour, one can reuse (i.e. "copy and paste") the hexadecimal value of that colour (here "0000FF") for any other element or attribute where one would want to apply the exact same colour.

Also note that, as far as general formatting is concerned, the font face “Courier New” was selected here, together with “Underline”. See Figure 10 for an example of the result in the Preview Area.

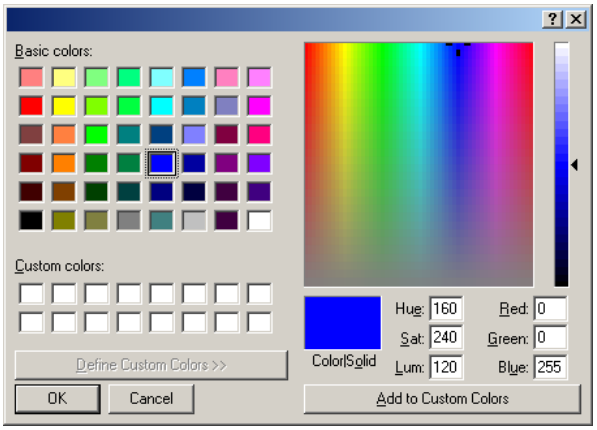


Figure 47: *Colour chart for selecting foreground and background colours of elements and attributes*

Text Before/After Options

In addition to formatting options such as bold and italics, and the selection of colours, common text/punctuation that should appear before or after specific elements or attributes can be configured with the style. This can be used, for example, to precede every usage note with the text “Usage:”, or to place all part-of-speech fields between brackets. By using styles, this sort of text/punctuation may be changed throughout the entire dictionary at once, simply by changing the style. The user is also freed from entering such text/punctuation manually, over and over. Using the text before/after options thus saves time, ensures consistency, avoids typing errors, and provides extra flexibility.

Note that for each style, there are two “Text before” and “Text after” options. The first of these, labelled “uses style formatting”, will be output *with* the formatting options of the style (e.g. **bold**, **magenta**) applied. The other options, labelled “before style formatting” and “after style formatting” will be output before/after the formatting options of the style are applied.

For example, in Figure 46 the text “%nE-mail: ” has been typed into the “Text before (before style formatting)” input box. This means that each “Email” attribute will start on a new line (%n), and will be preceded by the text “E-mail:” followed by a space. The formatting “Courier New” and “Underline” will not be applied on this “Text before” section, however. (Note that the font size is taken from the parent, “Contact”, in this case “10 pt”.) Again, see Figure 10 for the result in the Preview Area.

Note: All **formatting markup characters** (%n, %r, etc.) can be used within the Styles System. For a list of the different markup characters, see the section Using Text Formatting Within Attributes, in the chapter on Editing Attributes.

Group Style Properties for Multiple Elements or List Items

Group style properties are edited in the bottom right area of the Styles/formatting tab, under the heading “Style properties for groups of multiple child elements or multiple list items”. These options allow one to specify the separator character to use between elements when there is a “group” of elements, that is, when an element in the document tree contains multiple child elements of that type. For example, one may specify here that multiple translation equivalents (“TE” elements) under each “Sense” element should be separated by commas, or that multiple senses under a lemma should be separated by semi-colons.

In addition to specifying a separator character, the group style options also allow one to configure text that should appear before or after the entire group of child elements of the same type. This can be used, for example, to precede every phrasal-verb section of an article with “PHR V”.

The group style properties are also used to specify before/after text and separator characters for “list” type attributes that allow multiple list items to be selected from attribute lists, such as the part of speech attribute of the lemma and sense elements. Note that the group style properties have no effect for other non-list attribute types.

Automatic Numbering

A comprehensive “auto-numbering” system allows automatic numbering to be assigned to *any* element type, by means of an “Auto-number (Integer)” attribute type. So, for instance, to number the various senses of a polysemous lemma automatically, one will first have to create an “Auto-number (Integer)” attribute for the “Sense” element, as shown below:

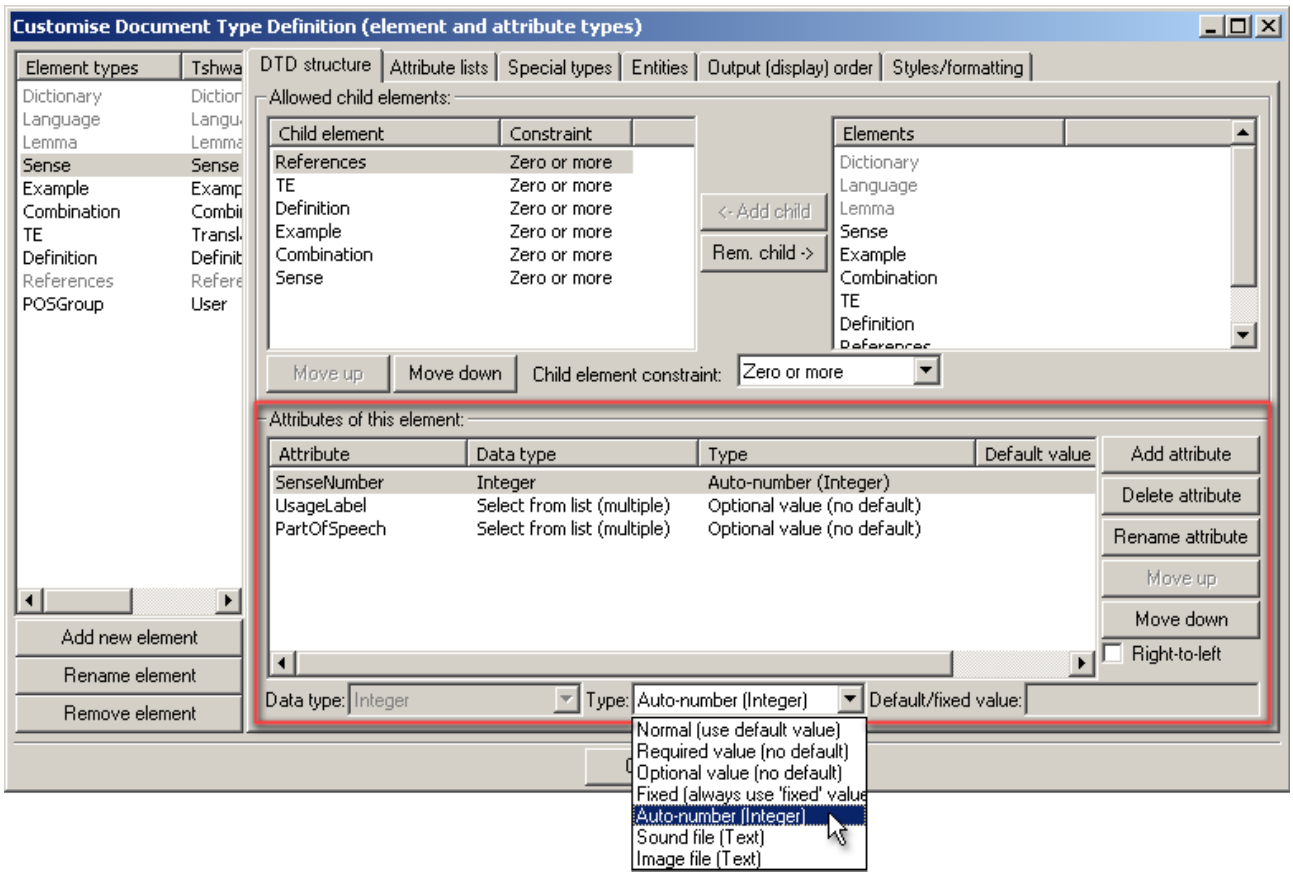


Figure 48: *“DTD structure” tab, Attribute editing window: Selecting the “Auto-number (Integer)” attribute type, used for automatic numbering of any elements*

The type of numbering scheme to be displayed in the output can be configured in the Styles System, via the “Numbering” button on the “Styles/formatting” tab.

So, for this example, if one first clicks on the “SenseNumber” attribute under the “Sense” element on the “Styles/formatting” tab, the following dialog will pop up:

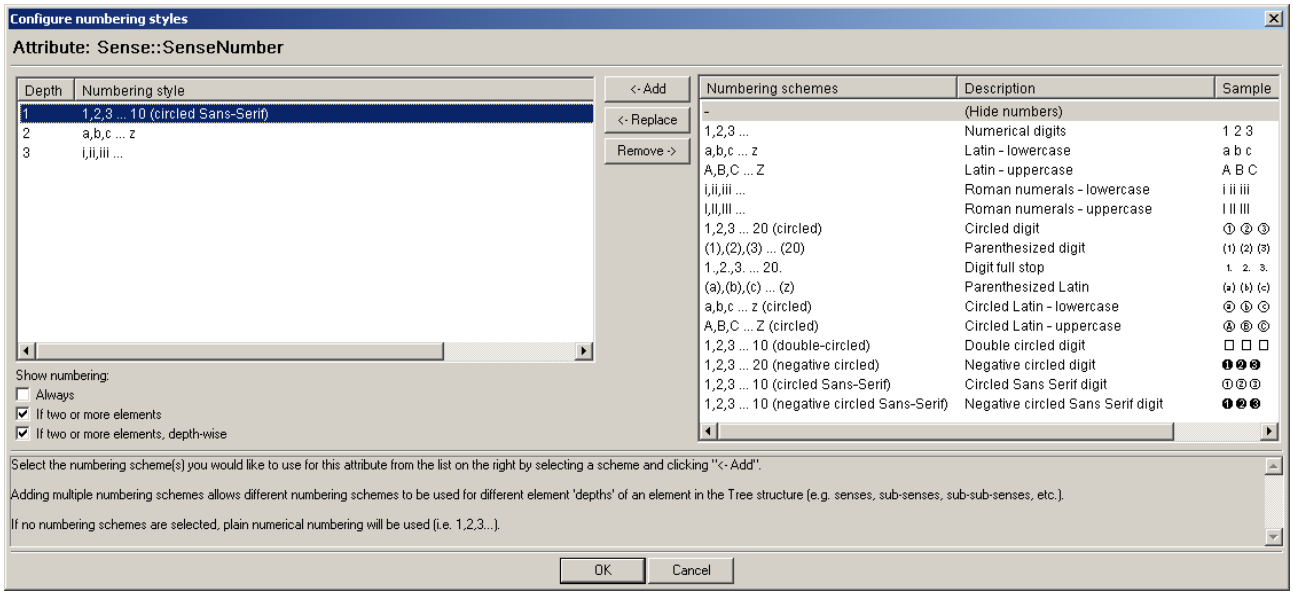


Figure 49: *Numbering styles dialog, used to select numbering schemes for "Auto-number (Integer)" attribute types*

The list on the right shows various numbering schemes, and clicking the "<- Add" button selects one of them. Selecting several (different) numbering schemes allows you to number the different Tree View levels of an element.

For example, if in a particular dictionary a sense can have subsenses, and these subsenses can again have subsenses, then one will need three levels of numbering. In the screenshot the "circled Sans-Serif digit" numbering scheme has been chosen for level one (see e.g. Figure 11), "Latin – lowercase" for level two, and "Roman numerals – lowercase" for level three. One can of course choose the same numbering scheme for all levels, but choosing different ones generally increases readability.

There are also options to always show the numbering (thus even if there is only one sense in a particular article, that sense will always be preceded with a number/letter "1"/"a"/"A"/"i"/...); or to only start numbering from the moment there are at least two items on the same level in a particular article; or to only start numbering when there are at least two items on different levels in a particular article.

TLex/tlTerm automatically recalculates the numbering for "Auto-number" attributes whenever you add, move or remove their associated elements in the Tree View.

For homonyms, plain numerical digits are used by default (see the DTD in Figure 31). If one would rather have homonyms 'numbered' as, say, "a", "b", "c", etc., then one can simply open the "Styles/formatting" tab in the Styles dialog, mark the "HomonymNumber" attribute, click on the "Numbering" button, and then select the "Latin – lowercase" numbering scheme. (Adding extra levels of numbering schemes will, of course, have no impact for homonyms.)

Note: For some numbering schemes to be displayed correctly, such as the "circled Sans-Serif digit", one will need to choose a font for the auto-number attribute (on the "Styles/formatting" tab in the Styles dialog) that supports it. (A font that supports all but one, cf. Figure 49, is "Arial Unicode MS".) In general, if you see **hollow squares** (□) it means that the selected font does not support the required Unicode characters. Please try to use a different font in that case.

Paragraph Style Options ("Indents and Spacing" Tab)

Several "Paragraph" and paragraph-related styles are available for each style in the Styles dialog under the "Indents and Spacing" tab.

If the "Paragraph" style checkbox is selected for an element/attribute, then that field will be output on a new line (in e.g. the Preview Area and formatted exports), using a new paragraph.

In the default styles in a new document, the main 'entry' element is selected as having the "Paragraph" option. Thus when generating (say) an RTF export, each entry begins on a new line with its own paragraph separator.

Paragraphs and Borders: The paragraph option is particularly useful when working with "box-like" fields, e.g. if you have a 'usage notes' box in a dictionary, or any other data field that should be surrounded by a box, as border styles applied to "paragraph" type fields generate a border around the entire paragraph (like the "Tip" box below).

Tip: When ticking the "Paragraph" option on a style, it is generally a good idea to also make sure a font, font size, color, and possibly background color are selected for that style. This is because (as paragraphs are not nested) paragraphs do not inherit these from parent elements.

Note that the "Paragraph" checkbox is also an 'obvious option' to use if you are literally creating, say, a "p" (for "paragraph") container element that is intended to structure paragraph content – e.g. similar to the "p" element in HTML.

Using Paragraph vs "%n" to Generate New-lines Before/Between Elements

The paragraph option is also useful when you want a certain field to always begin on a new line (e.g. if you always want sub-entries to appear on a new line, you could use the paragraph option).

An alternative is to use the "%n" markup symbol for new-line (though there are pros and cons to either approach).

The main difference between the two is that %n generates a "soft new line" (akin to e.g. pressing Shift+Enter in Microsoft Word or LibreOffice), while the paragraph option generates a "hard" new line – i.e. a full/proper new paragraph. Neither solution is "right" or "wrong", but each solution might be more appropriate in different circumstances. There are some differences, e.g. paragraph alignment, justification and spacing options only apply to entire paragraphs at once. The paragraph option also cannot currently be selected as applying 'only from the second instance of an element onwards' in a group of child elements, while the "%n" method can, as it can be specified in the "Text between" style field (in cases where you want the first of some element not to be on a new line, but the remaining such elements separated by new lines). See also the "[Advanced] Notes on Using Paragraph Styles" notes below.

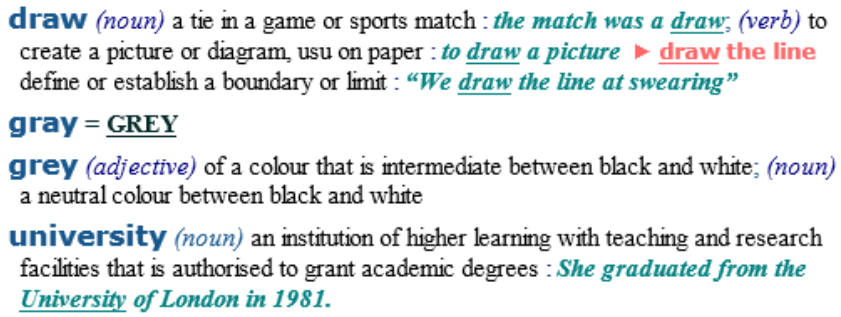
Paragraphs and "Show whitespace" Option

Note that the "**Show whitespace**" option (icon in toolbar that looks like a paragraph symbol, i.e. pilcrow "¶") will also display the two differently in the Preview Area – paragraphs (hard new lines) are displayed with the paragraph symbol (pilcrow "¶"), while "soft" new-lines (e.g. "%n" new-lines) are displayed as a left-pointing arrow.

Paragraph Margins (Spacing)

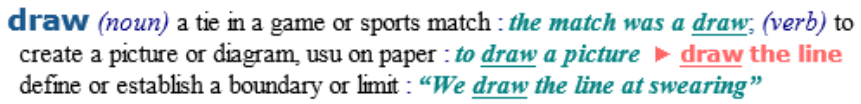
The options ‘Indentation before text’ (left-of), ‘Indentation after text’ (right-of), ‘Spacing above paragraph’ and ‘Spacing below paragraph’ can be used to set the left, right, top and bottom spacing around a paragraph-styled field.

Note that specifying some spacing either above or below the main Entry/Lemma element in a dictionary document can be very useful for creating a small visual “gap” between each entry, which helps the user to visually identify where entries start. An example of this is shown in this image:



Creating a Hanging Indent

A “hanging indent” can be created for a paragraph/entry by specifying a negative “First line” indent (e.g. -10pt), and a correspondingly-sized positive “Before text” indent, e.g. “10pt”. The ‘hanging indent’ effect is shown in this image:



Creating Nested Indents

Nested indents can be created, such that a left-indent accumulates increasingly with each instance of a child element in a hierarchy.

Note that this option is to some extent specific to our application, and doesn’t necessarily correspond directly to style options in some output formats, so may not always render as expected in some export formats.

[Advanced] Notes on Using Paragraph Styles

One minor ‘quirk’ (this is a relatively obscure point) that you might want to be aware of when working with paragraphs, is that (because paragraphs can't truly be "nested"), when outputting a paragraph style, the application must (for technical reasons) end the current paragraph at the end of the element/attribute, and also, when a new paragraph starts, any existing paragraph must be closed automatically). So in the example of using “Paragraph” at sub-entries, if you have other attributes that are output after the Subentries, they can end up 'outside' a paragraph in the Preview Area, e.g.:

<paragraph>draw (v) - use pencil on paper to create images</paragraph>

<paragraph>draw the line (v) (idiom) - establish a boundary or limit</paragraph>
Etymology: From Latin ‘dro’

In the above simple example, the ‘Etymology’ ends up, in a sense, 'floating' outside the specified paragraphs, and thus it may not correctly inherit all desired styles anymore – thus you may need to re-specify e.g. colors and fonts and font sizes for these attributes if you have such ‘floating’ cases. Also, paragraph elements should declare (in the Styles) at minimum a font, font size, background and foreground color (as these basic styles "reset" on each new paragraph in HTML).

Note that the above ‘quirk’ is not really harmful, but you may see minor side effects such as the color of some information (e.g. Etymology above) turning blue in the Preview Area when hovering the mouse over it – this is caused by failing to e.g. inherit a color.

Output (Display) Order

Element and Attribute Output (Display) Order

The *output (display) order* of elements and attributes in the Preview Area (and formatted output such as RTF, HTML, etc.) is configured *separately* to the DTD structure. This allows some separation of the underlying data structure and the final visual output. The output order of elements is configured using the “Output (display) order” tab in the Styles dialog (or the DTD editor dialog).

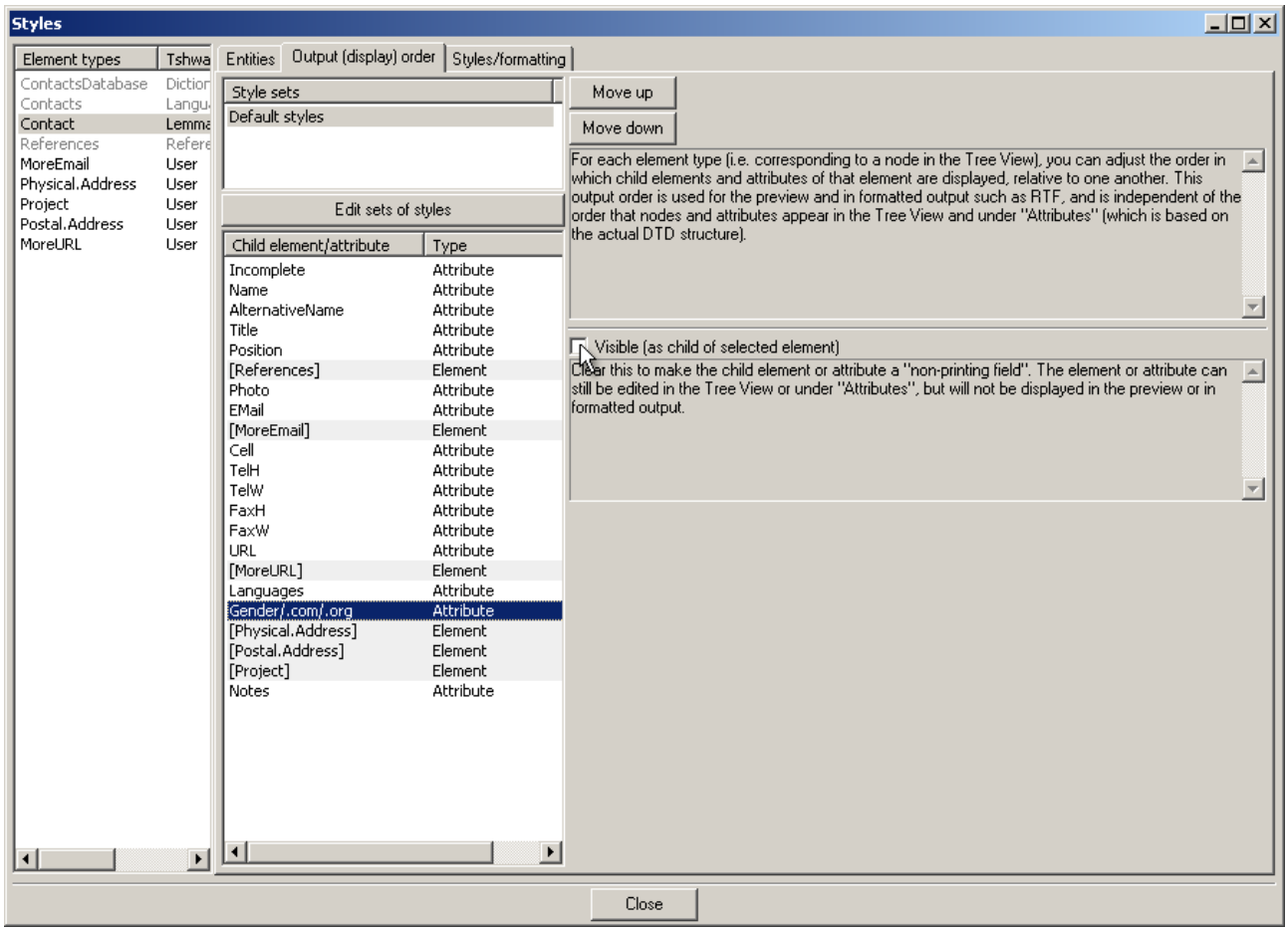


Figure 50: *“Output (display) order” tab in the Styles dialog:*
Configuring the output order of child elements and attributes of the “Contact” element

For each element type, one can configure the output order of (a) *child elements* of that element type, and (b) *attributes* of that type, all relative to one another. As an example, consider the "Contact" element type in the above screenshot, which has a number of attributes such as "Name", "Photo" and "Notes", as well as a number of child elements such as "References" and "Physical.Address". Some of the contact attributes (such as "Name") should appear *before* the "References" and "Physical.Address", other contact attributes (such as "Photo") should appear *after* the "References" but *before* "Physical.Address", while the "Notes" attribute should even appear at the *very end* of the entry.

When the "Contact" element is selected (from the list on the very left of the "Output (display) order" window), the list of all child elements and attributes of the "Contact" element is shown, and the output order of these can be modified by using the "Move up" and "Move down" buttons. One can thus specify that for the "Contact" element, the name should be displayed before any child references of the contact, that the photo should be included between the references and the child addresses, but that the notes attribute should be displayed at the very end of the entry.

Observe that, like the styles, changes made here are applied immediately in the Preview Area.

"Visible" Flag (Non-Printing Fields)

Each child element or attribute in the output order configuration may also be selected to be hidden in the Preview Area or output. This can be done by toggling the state of the "Visible" checkbox. This allows fields to be marked as "non-printing fields", i.e. fields that should not appear in the dictionary output. In the above screenshot, for example, the "Gender/.com/.org" attribute has been marked as a non-printing field.

When a child element or attribute is hidden, it *will still* be shown in the Tree View or under "Attributes (F1)", and it can still be edited – it is only disabled in the *output*. See in this regard Figure 10.

Note that it is also possible to quickly toggle an *element* type as visible/invisible by right-clicking on the node in the Tree View and selecting "Modify styles/Toggle visible" from the menu.

Entities

Entities can be used to define shortcuts to common text. These are then automatically substituted by the entity's associated corresponding text value in the output wherever the entity occurs in the dictionary. Entities can be used to enter characters not usually easily accessible on a keyboard (e.g. you can enter "©", and the software will automatically replace this when creating the output with the copyright symbol "©"). Entities are entered by typing an "&" character, followed by the character name, followed by a semi-colon. Examples of other entities are "£" for the pound symbol "£", and "…" for the ellipsis character "…".

You can view the list of entities, or add or remove entity types, using the "Entities" configuration dialog, available by selecting the "Entities" tab at the top of the DTD editor dialog ("Dictionary/Customise DTD (dictionary grammar)").

By default, entities are automatically substituted with their associated values in the Preview Area. This can be toggled under "Format (F4)" with the "Expand entities" checkbox.

Using Entities to Embed Labels within Other Fields and Further Customisation

Entities can also be used for situations where labels need to be embedded within other fields, as is the case for, for example, the French 'f' ("feminine noun") and 'm' ("masculine noun") gender labels. In a bilingual English – French – English dictionary one may for instance find the following entry:

contrivance *(noun)* ① [*tool*, *machine*] appareil^{m}, machine^{f}; ② [*scheme*] invention^{f}, combinaison^{f}

In an electronic environment, one would want users who click on the 'f' and 'm' gender labels to be sent to the correct meanings of those labels, rather than to random abbreviations. In existing electronic English – French – English dictionaries it is not uncommon to click on the 'f' and 'm' gender labels, and to be presented with an array of possibilities. Following up on each of them, may lead to:

- Clicking on 'f' may lead to:
  - **F, f** ... *nom masculin* ... F, f
  - **F** ... *abréviation* ... F, fr ... F

- Clicking on 'm' may lead to:
  - **M, m** ... *nom masculin* ... M, m
  - **me, m'** ... *pronom personnel* ... me; myself
  - **m** ... *abréviation* ... m
  - **m' ► me**
  - **M.** .. *abréviation* ... Mr

In other words, if one does not already know what these abbreviations stand for, one receives *no* guidance at all, with the first option for 'f' as "nom masculin" definitely confusing.

To avoid such problems, the 'f' and 'm' gender labels could be defined as entities (e.g. '&f;' and '&m;' respectively) which are replaced in the output with labels configured in a *single, central place*, i.e. on the "Entities" tab, as shown below:

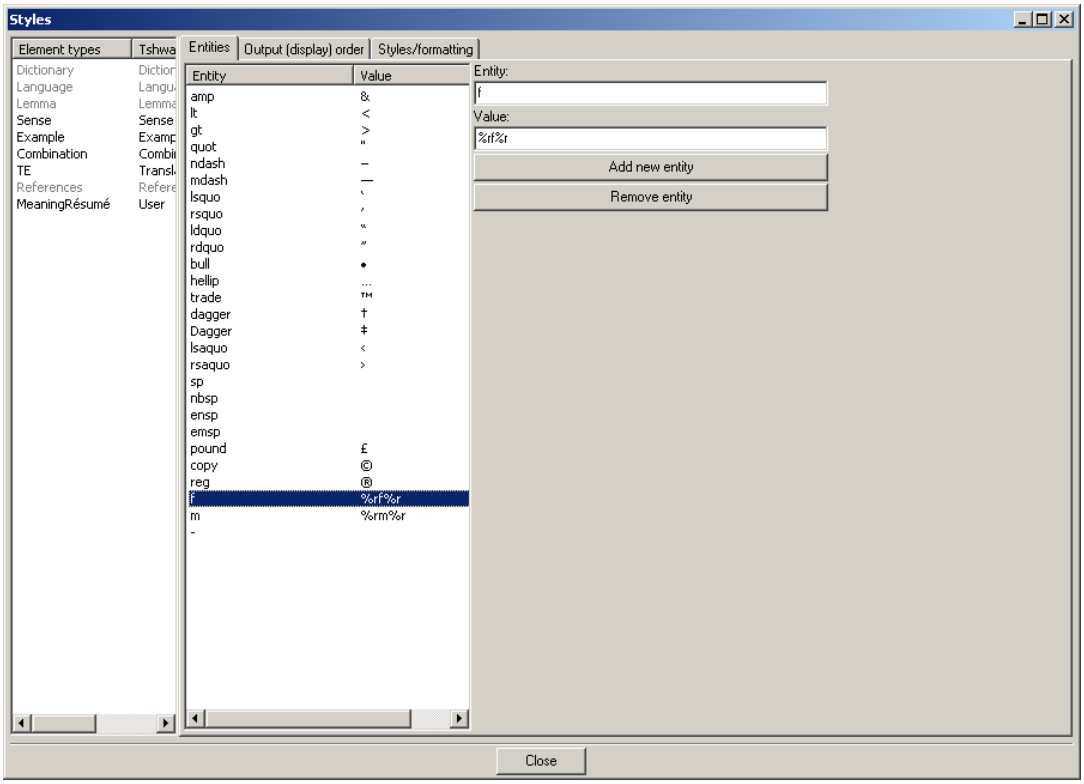


Figure 51: *"Entities" tab in the Styles dialog: Adding new entities for the 'intelligent' use of gender labels, in this case the raised 'f' for "feminine" and the raised 'm' for "masculine" [with '%r' for "superscript"]*

In the next screenshot, the use of the 'f' and 'm' gender labels as entities is illustrated.

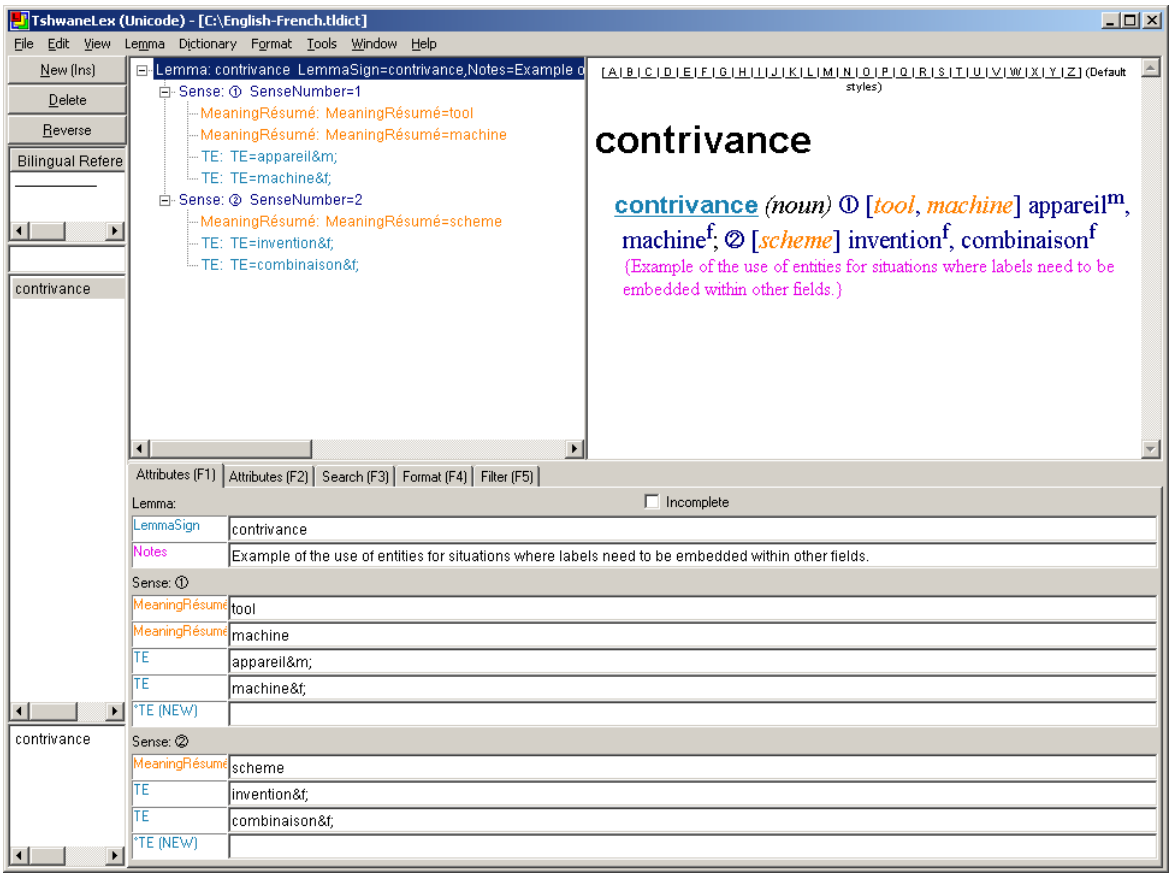


Figure 52: *Example of the use of entities (here '&f;' and '&m;') for situations where labels need to be embedded within other fields*

Using entities for such cases allows the software to be *aware* that these labels are part of the metalanguage, and to thus provide a more meaningful response should the user click on them. A powerful *extra mechanism for customisation* is moreover provided, as the language/appearance of the labels can, again, easily be adapted in one single place.

Style Sets, Or 'One Database, Many Dictionaries'

Not only can the dictionary grammar for any project be flexibly configured and then kept under control with the customisable and multilayered DTD editor dialog, given that all elements and attributes are also linked to a comprehensive Styles System for generating the output (and preview), one single database can efficiently hold several dictionaries. This outcome has been taken one step further, and functionality has been added with which it is possible to set up **multiple sets of styles** and to toggle between them. There is no upper limit as to the number of sets of styles that can be defined.

Broadly speaking, this is achieved by doing two things. Firstly, by making use of multiple element "categories" to which the various data attributes are assigned by the user depending on which dictionary or dictionaries they should appear in, and secondly by defining a different set of styles for each 'view' of the database, i.e. for each dictionary. Certain element categories are made visible or invisible in each style, which thus effectively functions as a kind of 'mask' that filters and reveals only the portions of data to be shown for the current dictionary. Additionally, this also allows a different 'look' to be defined for each dictionary.

In other words, the multiple sets of styles available in a *single* database, allow for the *simultaneous* compilation of a *multitude* of dictionaries: 'pocket' vs. 'full' editions, 'desktop' vs. 'electronic' editions, 'monolingual' vs. 'semi-bilingual' vs. 'bilingual' vs. 'multilingual' editions, etc.

Given that both the styles/formatting and the output (display) order can be manipulated for each "style set", both the "Styles/formatting" and the "Output (display) order" tabs of the Styles editor ("Format/Styles...") have a "Style sets" section in the top-left corner, as shown on the left screenshot below:

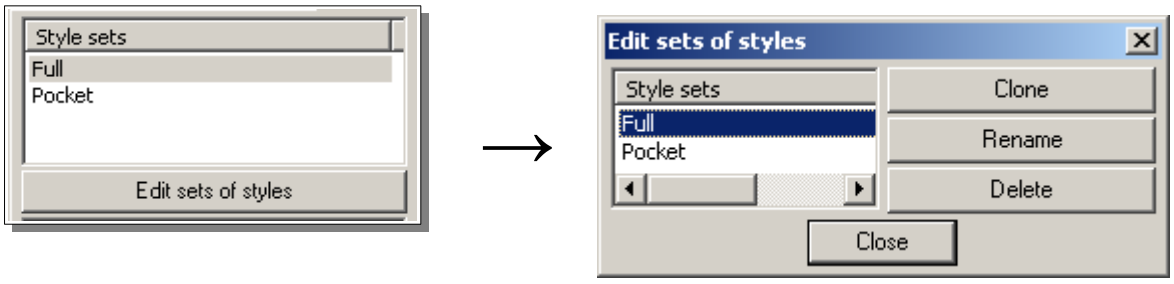


Figure 53: *"Edit sets of styles" under the "Format/Styles..." menu option*

Clicking on the "Edit sets of styles" button brings up a new dialog, shown on the right screenshot above, where a new style set can be configured starting from a clone of an existing style. Given a detailed-enough DTD, all the features discussed in the present chapter can be applied in creating multiple views. A sample file demonstrating a full vs. a pocket edition is available in the "Projects/Samples" folder of TLex. Please refer to that file. The settings illustrated there result in the following views of the same database:

- **Full edition:**

draw |drô| *(noun)* a tie in a game or sports match: ***the match was a draw***; *(verb)* to create a picture or diagram, usu on paper: ***to draw a picture*** ► **draw the line** define or establish a boundary or limit: "***We draw the line at swearing***"; *(verb intransitive)* to proceed or move steadily: ***a ship drawing near the shore***

gray = grey

grey |grā| *(adjective)* of a colour that is intermediate between black and white; *(noun)* a neutral colour between black and white

u·ni·ver·si·ty |yū'nə-vûr'sĭ-tē| *(noun)* an institution of higher learning with teaching and research facilities that is authorised to grant academic degrees: ***She graduated from the University of London in 1981.*** [Middle English *universite*, from Old French, from Medieval Latin *ūniversitās*]

- **Pocket edition:**

draw *noun.* a tie in a game or sports match *verb.* to create a picture or diagram, usu on paper • ***draw the line*** *define or establish a boundary or limit*

gray = grey

grey *adjective.* of a colour that is intermediate between black and white *noun.* a neutral colour between black and white

university *noun.* an institution of higher learning with teaching and research facilities that is authorised to grant academic degrees

With Ctrl+P (or "Format/Toggle current style set") one can toggle between the different style sets. If one has prepared, say, five style sets, then one can 'run through' all of them by pressing Ctrl+P consecutively.

"Masks" (Selectively Hiding Specific Information on a Per-Edition Basis)

(Please refer to the sample "Multiple Editions Sample using Masks.tldict".) This sample demonstrates the "Masks" aspect of the "multiple dictionaries from one database" functionality in TLex.

The database contains two 'editions' - a 'full edition', and a smaller 'pocket edition' derived from the full edition. One *entry* is used to generate two possible *output* versions (i.e. the two editions constitute two different "views" of the same underlying entry data). Each edition has its own set of styles, and thus its own 'look'. The styles also define whether or not particular fields are entirely hidden in an edition; for example, the "Pronunciation" field is hidden in the Pocket edition.

You can press Ctrl+P (or use "Format/Toggle current style set") to switch between the two 'views' (editions), or click on the name of the edition in the top right of the Preview Area.

There are generally two different 'methods' in which fields can be selected to be hidden for each edition. You can use either one method or the other, or both methods combined, depending on your requirements.

The first method (‘**visibility flag**’) is for if you want to hide ALL instances of a particular field throughout the dictionary for a specific edition. For example, the "Pronunciation" field is always hidden in this sample for the Pocket edition only. This method does not require the "Masks" feature. To do this, go to "Format/Styles", select the "Output (display) order" tab along the tab, select the "StyleSet" corresponding to the 'edition' (you need at least one StyleSet per edition), then, select the (parent) element of the field you want to hide at the left, then select the field (attribute) to be hidden in the "Child element/attribute" list, then tick or untick the "Visible" checkbox as desired.

The second method (‘**Masks**’) is if you want to selectively hide SOME instances (within entries) of a particular field for a specific edition only, e.g. if you want certain usage examples visible in the Pocket edition, but you want other usage examples to be hidden. This can be done using the Masks feature of TLex, for any element type, and is demonstrated in this sample on the "Sense" element, to exclude chosen senses from the Pocket edition. In the "draw" entry, for example, sense 2 has been 'excluded' from the Pocket edition - merely by ticking its "Sense::Exclude" box under "Attributes (F2)". (This requires the "Masks" to be configured first - this is explained briefly below.)

Note that TLex also 'intelligently' recalculates the automatic numbering for elements that are "masked out" - in this example, sense 3 automatically becomes sense 2 in the Pocket edition.

Finally, as a third ‘method’, one can also choose “abbreviated” labels for the Pocket edition under “Format (F4)”. (Abbreviated sets of labels can be configured in the 'DTD editor' - see the User Guide section "Attribute Lists".)

Configuring Masks

Note you can repeat 'STEP 2' to 'STEP 4' here to set up Masks on however many elements you need - e.g. you could mask out usage examples on a per-edition basis, definitions, translation equivalents, senses, etc. In this explanation there are also only two editions, a 'Full' and 'Pocket' edition, but you can create as many editions as you like.

STEP 1: Create 'StyleSets' for each edition

- Open your document in TLex (or tlTerm, tlDatabase etc.)
- Go to 'Format / Styles' to open the Styles dialog
- Under the "Style sets" list, click "Edit sets of styles"
- If you already see a 'Pocket' styleset in this list (e.g. if you're embarking from the sample), then you can skip the rest of these instructions in "STEP 1".
- If you don't already have a 'Pocket' styleset in this list, then select an existing set of styles (by default there will usually be only one here, named 'Default styles'), and click "Clone"
- Enter the name 'Pocket', and click OK
- You should now have two style sets in the list, 'Default styles' and 'Pocket'
- Select 'Default styles' and click the 'Rename button', then change the name to 'Full', and click OK

You should now have two sets of styles, one for a 'Full edition' and one for a 'Pocket edition'.

STEP 2: Add an 'Exclude' attribute (field) that will be used to actually select which information to 'mask' out

- Go to 'Dictionary / Customise DTD'
- Select the element at the left for which you want to be able to selectively exclude instances (e.g. 'Sense', or e.g. 'Lemma' if you want to selectively exclude entire entries)
- Click the "Add attribute" button at the right

- Enter the new attribute name "Exclude", or "ExcludeFrom" (this name can be anything, but 'Exclude' or 'ExcludeFrom' are good, as they help describe what it does) and click OK
- Make sure the Exclude attribute is selected in the list of attributes, and then (at the bottom of the dialog) change its "Data type" from "Text" to "Select from list (multiple)"
- You will get a warning about changing the attribute type - make sure you've selected the new Exclude attribute, then click 'Yes'
- You will get a prompt about creating a new list. If you're setting up masks for the first time, just click "OK" here to "Create new list". (If you've already been through these steps to create an Exclude list, then you can just select that existing list here.). The list will contain a list of 'editions'.
- Now select the "Attribute lists" tab at the top of the DTD customisation dialog
- Select the new "Exclude" list in the list of lists. With the "Exclude" list selected, do:
- Click the "Add new list item" button. Enter new list item name "Full" and click "OK"
- Click the "Add new list item" button. Enter new list item name "Pocket" and click "OK"

You should now have a new F2 checkbox list attribute named 'Exclude', with two list items for its list - one for each edition, i.e. 'Pocket' and 'Full'.

Note that the "Masks" do not work by selecting what we want to include, but by selecting what we want to exclude/hide for a given edition - hence the name 'Exclude'.

STEP 3: Set up the actual 'Masks'

This step basically 'links' each edition's 'style set', to its corresponding edition list item in the 'Exclude' list, as a TLex 'Mask'. (Don't worry if you don't fully understand what is happening here - just try follow the steps closely, and thereafter you should be able to just tick the excluded information under 'Attributes (F2)' while editing.)

- Go to 'Format / Styles'
- Select the 'Masks' tab at top right of the Styles dialog
- Select the 'Pocket' styleset under the list titled "Style sets"
- Click the "Add" button below the list
- Select the "Exclude" list from the 'Select a list' dialog that appears
- Then on the right hand side area of the Masks configuration window, it will now display a list of the items in the Exclude list, namely 'Full' and 'Pocket', with checkboxes at each - tick the 'Pocket' checkbox (meaning, for the 'Pocket' style set, we want exclude i.e. 'mask out' elements/entries where 'Pocket' is ticked under F2)

- Now select the 'Full' style set under the list titled "Style sets"
- On the right hand side area of the Masks configuration window, tick the 'Full' checkbox (meaning, for the 'Full' style set, we want exclude i.e. 'mask out' elements/entries where 'Full' is ticked under F2)

- Click the "Close" button at the bottom of the "Styles" dialog

STEP 4: Hide the 'Exclude' attribute from the Preview Area and output

Since the newly created 'Exclude' attribute at "STEP 2" is also just a 'normal' field, it will by default be visible in the Preview Area and in the exported output. This is usually not desirable. So for the final step, we just need to hide this attribute.

The attribute hidden/visible state is chosen separately for each 'set of styles', so we need to do it twice, once for the 'Full' edition style set, and once for the 'Pocket' edition style set.

- Go to 'Format / Styles'

- Select the 'Output (display) order' tab at the top

- Select the 'Full' style set under 'Style sets'
- Select the 'Lemma' element in the list at the left (or 'Sense', or whichever element it was you selected at STEP 3)
- Select the 'Exclude' 'Attribute' in the 'Child element/attribute' list
- Untick the 'Visible' checkbox

Then, repeat for the 'Pocket' style set:

- Select the 'Pocket' style set under 'Style sets'
- Select the 'Lemma' element in the list at the left (or 'Sense', or whichever element it was you selected at STEP 3)
- Select the 'Exclude' 'Attribute' in the 'Child element/attribute' list
- Untick the 'Visible' checkbox

Finally, click the "Close" button at the bottom to close the Styles dialog.

You can now start masking out information on a per-edition basis as you edit. Remember the toggle option 'Format / Preview all sets of styles' to enable/disable real-time preview of ALL editions, and you can use the Ctrl+P 'Format / Toggle current styleset' command to switch between edition 'views' (or select the current edition from the top right of the Preview Area).

“Smart Styles” (Dynamically Customisable Styles) [Advanced]

TLex/tlTerm allow you to set up rules (of any possible complexity) that allow the visual appearance (style) of a field to change dynamically in different situations. As a simple example, you might want the automatic punctuation before a usage example to change depending on the field that happens to precede it. Or, you might want to automically generate a fullstop after the example only if it doesn't end on an exclamation or question mark.

All of these things can achieved by attaching a “Lua modifier script” to a style. This can be done under the “Advanced” tab in the Styles dialog. There are several samples included with TLex demonstrating the “Smart Styles” functionality. The screenshot below shows one of these samples, in which the top 500 most frequent headwords (based on corpus rank) are highlighted in red and with an asterisk.

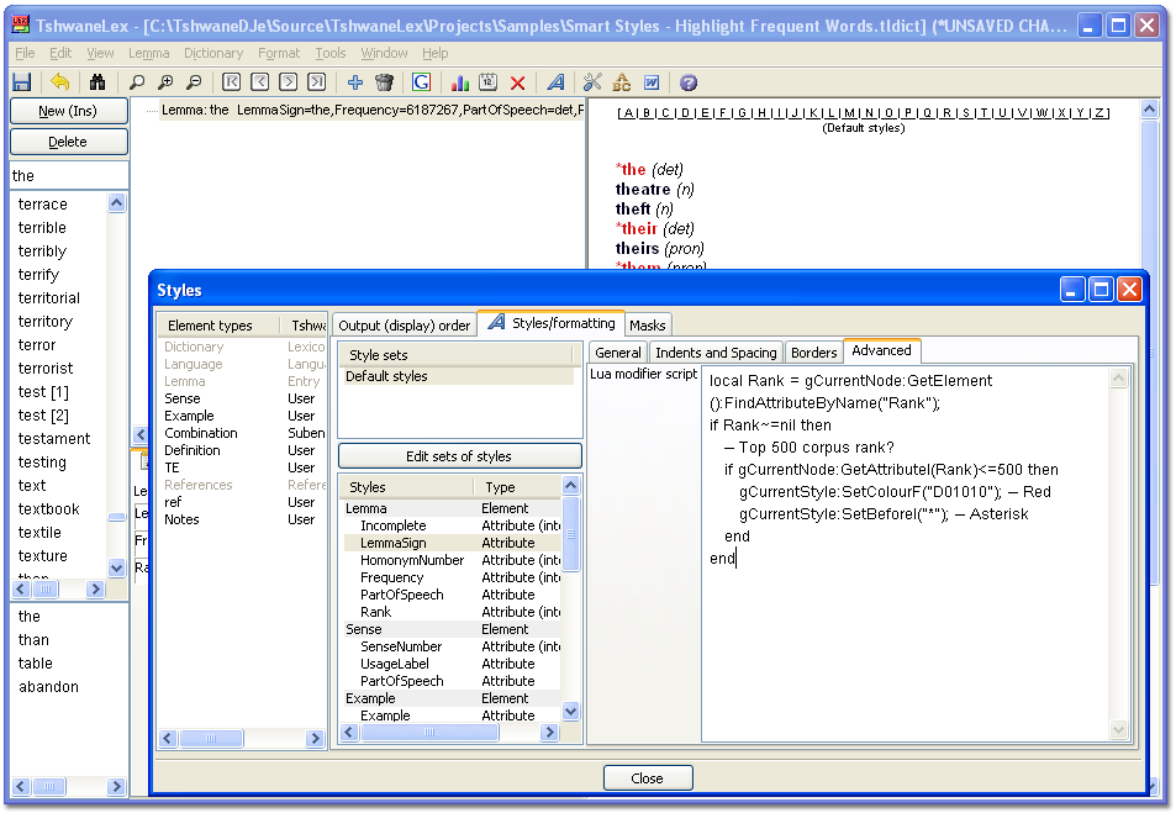


Figure 54: Automatically and fully dynamically highlighting the most frequent words (red, and with a "") based on the corpus frequency ranking. (Data shown in sample is the Adam Kilgarriff BNC top 6318 wordlist.)*

Creating a Thesaurus

TLex can be used to create a thesaurus database. For more information, please see the “Thesaurus” sample file included with TLex

Multi-Document Smart-Search

The 'Multi-Document Manager / Smart-Search' (added in version 2018) allows you to load multiple files/documents in various formats into the ‘background’, and then the Word/Excel search widget can auto-search in these files too, in addition to your own currently open tlTerm/TLex document. You can also directly do searches in the Multi-Document window itself.

You can load either your own documents (e.g. tlTerm or TLex files), or third party documents such as downloadable TBX, TMX or XLIFF files.

You may continue viewing or editing your own main document ‘as usual’ while the Multi-Document Search window is open. This may help, for example, to use TBX/TMX/XLIFF files as search resources while doing lexicography or terminology work.

Video briefly showing Multi-Document Smart-Search in action:

https://youtu.be/Sj4fb6lGkGc?list=PL_xDVMreHQV13lJiTx6h0L04j_OiTUcDv&t=505

This tool is accessible from within the Word/Excel search widget pop-up window, or from the application 'File' menu, from the application 'Edit' menu, from the main application toolbar, and more.

Supported file formats currently include tlTerm files, TLex files, TMX (Translation Memory eXchange), TBX (TermBase eXchange), and XLIFF (XML Localisation Interchange File Format).

This also allows the Word/Excel search widget to effectively search *multiple* tlTerm files at once, as well as other formats at the same time. (Please note that for .TMX files, you may currently need to copy the relevant TMX DTD file into the folder containing your TMX files if you get DTD-related error messages.)

The “General text commands” menu is also available when clicking on search results in the Multi-Document window. This allows you to, for example, do things like send terms/translations in search results from a TMX or TBX file directly to e.g. Word/Excel with commands like ‘Send to Word’ and ‘Send to Excel’. See the chapter on the Word/Excel search widget for more information about this menu.

This tool can also be used to ‘batch import’ data from a TMX or XLIFF file into tlTerm – see the sections on importing data into tlTerm for more information.

Import Wordlist / CSV (Comma Separated Values)

Many new entries can be added to the dictionary at once using the "File/Import/Wordlist or CSV (Comma Separated Values)" menu option. This imports data in one or more columns (one entry per row) from a CSV file saved from spreadsheet software such as OpenOffice.org or Microsoft Excel. Each column can be "mapped" to a certain attribute in the DTD while importing.

In order to assign attributes to columns, one must **double-click** on them in the left-side list of Element::Attribute names. The same attribute can be added more than once. To remove an assigned attribute, double-click on it in the right side list.

Apart from the addition of batches of new lemma signs to the dictionary, this function can thus also be used to add corpus-derived frequencies to existing lemmas, or to add any other corpus-derived or otherwise available layer(s) of data. Below, an example is shown of the import of three columns of data from a CSV file:

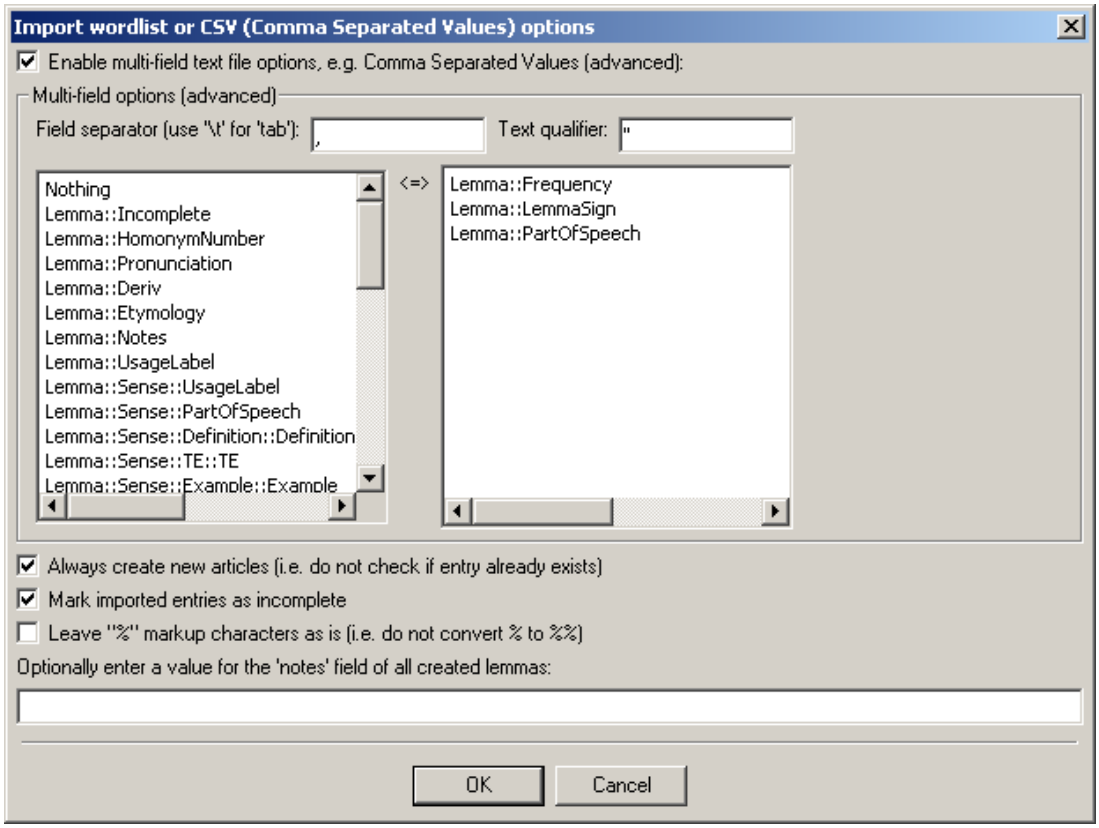


Figure 55: *Import wordlist or CSV (Comma Separated Values) options dialog, here used to import three columns of data (frequencies, lemma signs, and parts of speech)*

In the above example, a new entry will be created for every "lemma sign / part of speech pair" (cf. the "Always create new entries" option), and all imported data will be marked as incomplete. Importing data can be repeated as often as one wishes, and this at any stage during compilation.

Note: The "Import Wordlist/CSV" tool correctly handles **Unicode** files (particularly if saved in a format like "utf8 with signature", see Character Encoding Issues below), and includes an option to

allow for the use of the **special formatting markup characters** (%i, %k, etc.) in the source text/CSV data. For the latter, see the section Using Text Formatting Within Attributes, in the chapter on Editing Attributes. Also note that when importing several columns, the data can immediately be placed in the correct fields of the **DTD**. For the latter, see the chapter on Customising the Dictionary Grammar using the DTD.

Note also that some software, such as older versions of Excel, are not able to save Unicode/UTF8 format CSV files.

Character Encoding Issues

⚠ Pay particular attention to characters with diacritic markers or accents (e.g. "ê", "é") when importing data, and double-check they import correctly.

If you find that some (or all) characters don't import correctly, please try open the CSV file in a normal *text editor* (e.g. Windows Notepad), and re-save it with encoding settings as "UTF-8 with signature" (also called "UTF-8 with marker" or "UTF-8 with beginning of file marker" or "UTF-8 with BOM"), then try import it again.

The reason for this is that some versions of software like Excel may export CSV files in a format lacking this special "marker" which TLex/tlTerm/tlDatabase uses to recognize the text as 'standard' UTF-8-encoded Unicode. Without this marker, TLex/tlTerm/tlDatabase may try read the file using your system's default encoding (typically some legacy encoding, e.g. Windows-1252), which will produce the wrong results if it's a UTF-8 text file.

You should also be able to use 'UTF-16' Unicode text formats.

Importing XML [Advanced]

Data in XML form can be imported via the "File/Import/XML" menu option (provided, to a reasonable degree, that it is well-structured).

It is possible to both import “clean” (i.e. as a new, blank document), or do a “merge” import into an existing database. It is usually best to import data into a 'clean/empty' document, i.e. to select the XML import command when no database is open in the software.

NB: After importing XML, there would usually be no styles, thus all imported entries will usually be displayed in a default text style in black on a white background. You can use the "Format/Styles" menu option as usual to add styles once you are satisfied with the import.

TLex has one or two basic 'expectations' of how the data should be structured in order to import the data in a meaningful way (i.e. in a way that allows TLex to 'understand' what some of the key fields are, such as the headword). The following is an example of roughly the simplest XML document that can be thrown at the importer:

```
<Dictionary>
      <Language>
            <Entry LemmaSign="cow">
            </Entry>
      </Language>
</Dictionary>
```

or just:

```
<Dictionary>
      <Language>
            <Entry LemmaSign="cow" />
      </Language>
</Dictionary>
```

Note that the element for a 'dictionary entry' appears at the third depth level in the document, and should contain an attribute called "LemmaSign" that contains the headword; this allows TLex to recognise which attribute it should use as the headword for purposes of sorting, indexing in the Lemma List, and so on. (If the headword is in a different element or attribute, it will still be imported - TLex will just not 'know' to use that field for the Lemma List and so on.)

The names of the elements above ("Dictionary", "Language" and "Entry") can be anything, although their structure is important (i.e. second-level element represents each 'section' or 'side' of a dictionary within TLex, and third level represents the list of entries within that section).

Note that you do not necessarily need a DTD attached to the data - if importing XML data with no DTD, TLex will attempt to construct a DTD based on the elements/attributes it encounters. For well-structured data, this can work well.

Here is a slightly more complex example (note "TE" stands for "Translation Equivalent" for, in this case, bilingual English - Afrikaans data):

```
<Dictionary>
      <Language>
            <Entry LemmaSign="dog">
                  <Plural>dogs</Plural>
```

```
                  <Sense>
                        <TE TE="hond" />
                        <Definition Definition="A domestic mammal that barks" />
                  </Sense>
            </Entry>
            <Entry LemmaSign="cat">
                  <Sense>
                        <TE TE="kat" />
                  </Sense>
            </Entry>
      </Language>
</Dictionary>
```

The entries do not need to be correctly sorted within the XML (e.g. "dog" then "cat" above) - TLex will automatically resort them according to the default configured 'sort method' (which can also be changed at any time later on).

'Merge' XML Import

If you want to import entries into an existing database, the most important thing is to 'tell' the importer which 'side' (section/language) of a dictionary to import sets of entries into. This is done by filling in the language "Name" attribute with the exact same name configured for a language side/section under "Dictionary/Properties". This is shown in the following example:

```
<Dictionary>
      <Language Name="English">
            <Entry LemmaSign="dog">
                  <Sense><TE TE="inja" /></Sense>
            </Entry>
      </Language>
      <Language Name="Zulu">
            <Entry LemmaSign="inja">
                  <Sense><TE TE="dog" /></Sense>
            </Entry>
      </Language>
</Dictionary>
```

To do the 'merge' import, one then just selects "File/Import/XML" while the desired database is open.

NB: It is a good idea to always do a 'File/Create a backup' before doing a 'merge import'.

Post-Import Processing / Data Remodelling

For various reasons, it is seldom the case that freshly imported XML data can be used “as is”; one often has to do various kinds of clean-up, processing and data remodelling in order to make the data more useful. Depending on what one wants or needs to do, this can often be non-trivial, and involve e.g. specialised Lua scripts to manipulate the data. If you are in doubt, TshwaneDJe is highly experienced in dealing with these kinds of issues; contact the company to discuss your requirements.

“XML Line by Line” Importer

The “XML Line by Line” importer is a unique specialized, niche conversion tool that is especially suited for scenarios where many entries may contain bad XML - e.g. primarily data conversions from other formats (e.g. attempting to mash SGML or SGML-like data). This allows partial importing, even when potentially hundreds of errors are present that would ordinarily cause strict XML loading software to fail. The primary tenets of the “Line by Line” importer are, one line in the XML file corresponds precisely to one created entry, and each line contains only the elements that are *child nodes* of the main entry node.

NB: Depending on your data, this might be a poorly suited importer to use - there are several different potential methods in this software for importing XML data; it's important to use the correct one depending on your particular needs.

Please also see the “XML Line by Line import” **sample** that comes with the software, which contains additional information and a working sample that you can examine for guidance on how it would expect your data to look.

Note that this functionality is not available in the “old Document-Object-Model” version of TLex.

tlTerm: Importing Terminology from Other Formats

NB: Always create full, proper, extra backup copies of your existing data before attempting to bulk-import data into your document, and do test runs first on an extra copy, to make sure it imports the data the way you want it.

CSV (Comma-Separated Values) or Excel

In tlTerm, the "File / Import / Wordlist or CSV" or "File / Import / Excel" commands may be used to import column-based terminology data into tlTerm.

TBX (TermBase eXchange) Format

TBX is a format meant to help with exchanging terminology data between different terminology applications.

If you have data in TBX format (e.g. that has been exported from other terminology software), you should be able to use the "File / Import / TBX" command, and then save it as a tlTerm file. *(TBX files typically have a ".tbx" extension, and should be in a certain standard XML format.)*

If you have problems importing a valid .tbx file, please contact us and send a sample of the file.

Tip: There are useful TBX files available online that you can download, such as the IATE EU terminology, and the Microsoft Terminology Collection.

Important: You should always check the conditions of use and copyright/licensing terms when downloading a resource to see what you are permitted (by law, or conscience) to do with the data. (E.g. in many cases, you may not republish or redistribute someone else's work, and/or may only be allowed to use it for private use, e.g. to do searches if doing translation, or if researching a word or phrase.)

Tip: If you are using the 64-bit version of the software, it is possible to load *very large* TBX files, such as the IATE EU terminology, which is multiple gigabytes in size. The 64-bit version can use all available memory on your computer for massive databases.

See this video for instructions and a demonstration:

https://youtu.be/zsDLr3R6rbI

TMX/XLIFF (Batch Import using Multi-Document Tool)

To load TMX data *into tlTerm format*, create a new empty tlTerm document (or open an existing tlTerm document), then you may use "**File / Open for Multi-Document Smart Search**", select one or more TMX files, then right-click on the file name(s) in the Multi-Document Smart-Search dialog

and select "**Auto add all entries from selected document(s)**". (Note that not all fields may be imported. In future versions we intend to add more options for importing more fields.)

To just browse/view a TMX file *in* tlTerm, use "File / Open / TMX".

The same procedures should apply to XLIFF.

Video showing how to batch import TMX/XLIFF data using the Multi-Document tool:

https://youtu.be/Sj4fb6lGkGc?list=PL_xDVMreHQV13lJiTx6h0L04j_OiTUcDv&t=697

⚠ When working with data obtained from third parties (e.g. downloaded from the Internet), make sure you first check the conditions of use and licensing/copyright terms of that data regarding how you may use it.

Exporting Data from TLex/tlTerm

Copying Lemma Signs, Terms and Articles

Copy Lemma Sign / Term

To copy the lemma sign or term (of the entry you are working on) to the clipboard, from where it can be pasted to any other location (Ctrl+V), use the "Edit/Copy lemma sign" (Ctrl+Shift+C) menu option.

Copy Entry Text

To copy the full entry text (of the entry you are working on) to the clipboard, in text-only format, use the "Edit/Copy article text" menu option.

Copy Entry HTML

To copy the full text (of the entry you are working on) to the clipboard, in HTML format, use the "Edit/Copy article HTML" menu option.

Exporting the Database, in Part or in Full

Under the "File/Export" menu option, various ways to export your data are available. These are:

Lemma Signs [TLex] / List of Terms [tlTerm]

Use this to export only the list of lemma signs (or terms). The "Data export options" shown in the screenshot below are available for this type of export, meaning that filters can be used, and articles marked as incomplete can be included or subtracted. If you are working on a (semi-)bilingual dictionary, you will be prompted for a filename twice – once for each side of the dictionary.

There are various uses for a list of all the lemma signs / terms in your database, one of them being to build a crude wordlist-based spellchecker (e.g. for a language for which no spellcheckers exist as yet).

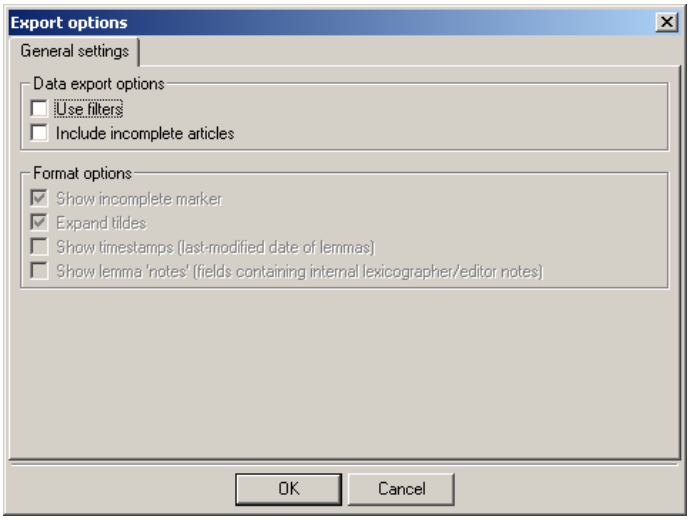


Figure 56: *Export options dialog, General settings: Data export options*

Text

Use this to export the text, in text format, of your dictionary or termbase. The general export options shown above are available. If you are working on a (semi-)bilingual dictionary, you will be prompted for a filename twice – once for each side of the dictionary.

A possible use for this type of export is with spellchecking purposes in mind. (Corrections will of course have to be made in the TLex/tlTerm files itself.) Note that if one is working with, say, a bilingual dictionary, and wishes to spellcheck the two languages separately, one can first use the Styles System – more in particular the "Visible" flag on the "Output (display) order" tab of the Styles dialog – so as to export only one of the languages at a time.

RTF (Rich Text Format)

This is the most common way to export data, as the text *formatting* (in black-and-white) is saved as well. See the section on Printing your Dictionary, in the chapter Getting Started with TLex: QuickStart Guide, for more information. Various format options, as illustrated below, are available:

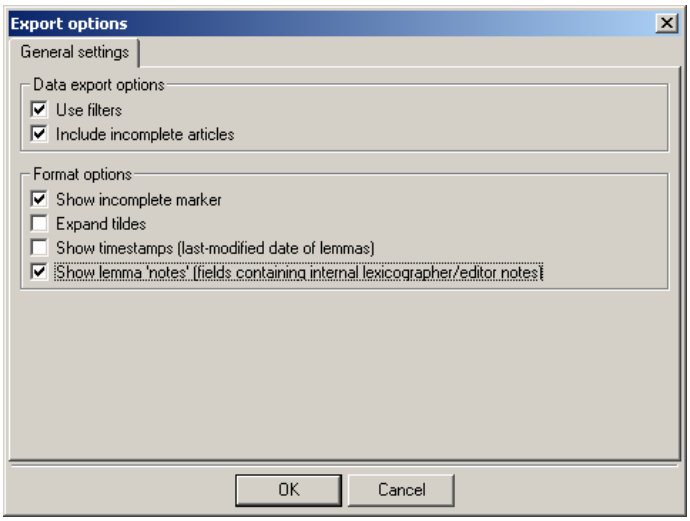


Figure 57: *Export options dialog, General settings: Data export and format options*

NB: If one has included images in the database, the RTF output and the image files must be stored in the same folder.

HTML (Web Page)

Use this to export the data, in HTML, when the intention is to place the result online as a web page, or as a set of web pages. In addition to all the general settings as discussed above, there are several HTML-specific options that are available.

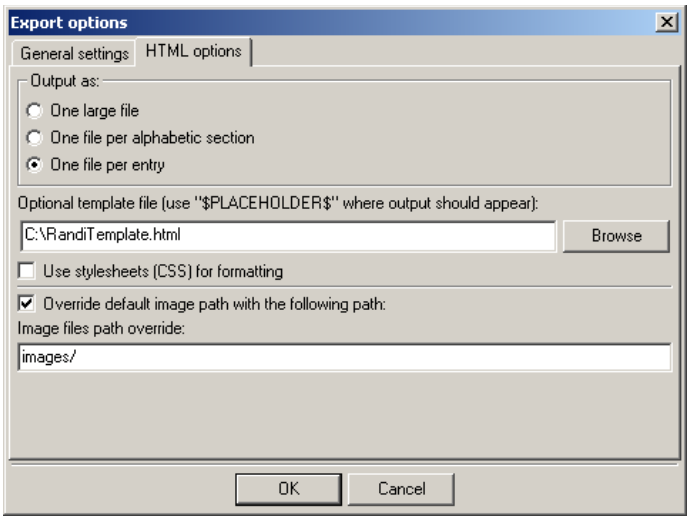


Figure 58: *Export options dialog, HTML options*

The first of the HTML-specific export options allows you to select to export the data either as one large HTML file, or to split the output up into alphabetic sections, or even into one file per article.

A “template HTML file” can optionally be specified, in which the output will be “dropped” at the “$PLACEHOLDER$” text.

Cascading Style Sheets (CSS) can also optionally be used.

Images are typically stored in an “images” folder online; use the relative path to it, to override the path stored under “Dictionary/Properties”.

If you are working on a (semi-)bilingual dictionary, you will be prompted for a filename twice – once for each side of the dictionary; and the chosen options (e.g. “one file per entry”) will be applied to the two sides. This thus means that one can end up with a very large number of files.

XML (eXtensible Markup Language)

Use this option if you want to export your data to the open, standard XML file format. In addition to the general options as seen in Figure 56, there are two XML-specific options. These two allow cross-references and list attribute values to be exported as text (mimicking more closely the formatted output), rather than by internal ID. One XML file is generated, even when saving data from a (semi-)bilingual dictionary.

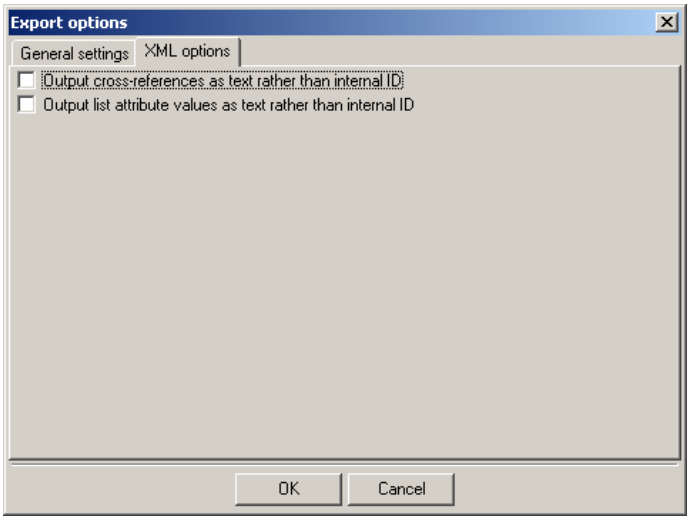


Figure 59: *Export options dialog, XML options*

XML (Formatted)

The plain “XML” exporter outputs your data in its *internal, structured* format. The “Formatted” XML exporter applies the basic Style and output order transformations to your data, and thus exports your data far more closely to how it appears in the *Preview Area*, but semantically tagged.

This exporter is most useful if preparing data for, for example, typesetting, as it can be brought into typesetting software and then easily styled, as each field is semantically marked up.

Note that, unlike the “plain” XML exporter, the “Formatted” XML output does *not* lend itself easily to being imported back into the software, as much of the *structural* information is lost.

Exporting to InDesign

It is possible to bring your data into Adobe InDesign various routes, including the HTML, RTF and 'XML (Formatted)' exporters. RTF is discussed below.

Exporting to InDesign via RTF

1.. [Optional] Try to get your styles as close to correct as possible in TLex/tlTerm already - this will minimize the amount of work that needs to be done later in InDesign
2.. [Optional] If you have multiple stylesets, select the desired styleset with Ctrl+P in TLex/tlTerm.
3.. Export RTF from TLex/tlTerm ('File/Export/RTF (Rich Text Format)').
4.. Create a 'new' document in InDesign, with the desired or estimated projected number of pages, and the desired number of columns.
5.. Select the 'File/Place' command, and select the RTF file that was exported from TLex/tlTerm.
6.. Hold in Shift and click in the top left of the first column. The entire document should now be imported.

Once in InDesign via RTF, each field (information type, i.e. element/attribute) has an associated 'style' which may be manipulated centrally in InDesign.

Exporting a Range of Entries or a Single Alphabetic Section, e.g. “A”

This can be achieved by “tagging” the range of entries that you want to export, creating a “tag filter”, and then exporting with 'Use filters' applied. See the “Tags” chapter for more information on how to do this.

Network (ODBC Database) Support [Advanced]

Overview

ODBC (Object Database Connectivity) is a standard interface for connecting **applications** to a **relational database server**, such as PostgreSQL, MySQL, Oracle, or Microsoft SQL Server.

The built-in ODBC support in TLex/tlTerm/tlDatabase allows a central database document to be viewed and edited simultaneously by multiple end-users. In this case, TLex/tlTerm/tlDatabase functions as a 'client', connecting to a *relational database server* that hosts the database content.

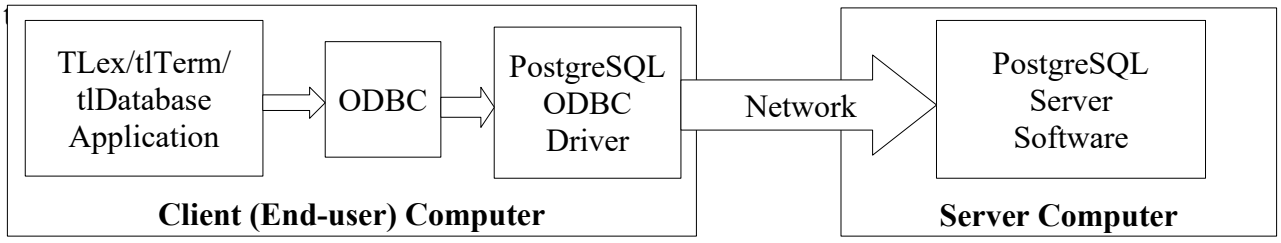
The relational database server software can be any of the major SQL servers, namely PostgreSQL (free/OpenSource), Microsoft SQL Server (commercial), Oracle DB (commercial) or MySQL (free/OpenSource). Apart from the relational database server, no other particular software is required on the computer designated to run as the server, and the server computer can have any Operating System that supports the selected relational database software (e.g. Linux as server is supported); TLex/tlTerm/tlDatabase don't need to be installed on the server computer.

From the point of view of the end-user, TLex/tlTerm/tlDatabase functions nearly identically in ODBC networked mode, as it does in stand-alone (normal file editing) mode; the main functional difference is the presence of an entry check-in/check-out system (see the "Entry Locking" section in this chapter) to prevent multiple users creating conflicting edits simultaneously on a single entry.

On the server computer, typically only the relational database server software itself is required to be installed and configured. On each client (end-user) computer, TLex/tlTerm/tlDatabase interfaces with the database server via an **ODBC (Object Database Connectivity) "data source"**. ODBC is built into Windows, and you can generally configure ODBC "data sources" via the Windows Control Panel. Thus on each *end-user's* machine is typically installed:

(a) the **TLex/tlTerm/tlDatabase application**, and
(b) the **"ODBC driver"** for the particular relational database software.

The general architecture is shown conceptually in the following figure, using the PostgreSQL database server as an example:



Troubleshooting

If you experience problems setting up ODBC, please also visit our Knowledge Base - there is some additional information there:

http://tshwanedje.com/kb

Configuring an ODBC Database

To initially save a new or existing project to an ODBC database, you must basically follow these steps:

1. Install the relational database server software (e.g. PostgreSQL) and configure the relevant security permissions to allow the client computers access. (The exact procedure for doing this is completely specific to the relational database server software that you are using; e.g. for PostgreSQL you may need to edit pg_hba.conf, and may need to configure your timezone correctly.)
2. Create one new, empty database in the relational database server software, with a single relational database username/password that has access to it (e.g. if using PostgreSQL, this step might typically be done using pgAdmin, though there are many different ways)
3. Install the ODBC client driver on your system
4. Create and configure a new ODBC "data source" on your system, using the abovementioned username/password in step 2 for access. In Windows XP, the data source can be set up by going to "Control Panel / Administrative Tools / Data Sources (ODBC)" (**NB:** For 64-bit Windows, see the notes below). [**NOTE:** As of April 2014, you can also use the "Tools / Open Windows ODBC Configuration Dialog" menu option within the application in Windows – this is the simplest way to always open the correct ODBC administrator window.]
5. Open TLex/tlTerm/tlDatabase on your system, and either load an existing document or create a new one (e.g. via 'File / New').
6. Select "File / Save as: ODBC database" from the menu, and enter the exact data source name you created under step 4 as the "data source". **You should only need to do this "Save" step once** – this initially saves your document into the relational database server and sets it up for subsequent access by all the other client systems. Note thatTLex/tlTerm/tlDatabase will prompt for two things here, before saving the database:
    1. ODBC data source name: The name of the configured ODBC data source.
    2. Table name prefix: All table names in the schema may **optionally** be given a prefix. This simply allows multiple different TLex/tlTerm/tlDatabase projects to be stored in one actual relational database on the server (this is sometimes useful in cases where, for example, the number of databases available is limited by a server hosting plan). Unless otherwise desired, it is safe to simply leave this on the default "tl_". (This field can also be used to save a separate "test" database for testing purposes, e.g. you could use a prefix "test_" for a testbed environment, and the default "tl_" for the real database.)

Depending on the database type and ODBC driver, you may also be prompted for the database username/password when saving or opening the ODBC data source.

If user management has not yet been set up for the project, TLex/tlTerm/tlDatabase will also display the user management configuration dialog during the above process, in which you may configure additional users. **Important: By default, a single "Admin" user is created with the default password "Admin"**. See the chapter on User Management for more information. Note that for ODBC databases, you cannot disable the integrated user management.

Once you have initially saved the ODBC database, you can open and work on it largely as if it were an ordinary TLex/tlTerm/tlDatabase file (though, of course, open it using "File / Open ODBC (with local cache)".

Configuring macOS MySQL Client (64-bit Sierra, Mojave, Catalina etc.)

Step 1: Download the "Stable Version" .pkg for Mac OS X here: http://www.iodbc.org/dataspace/doc/iodbc/wiki/iodbcWiki/Downloads and run and install it (you may have to allow to 'open anyway' under System Preferences → Privacy)

This will install the iODBC 64-bit ODBC administrator into your Applications folder.

Step 2: Go to this downloads area:
https://dev.mysql.com/downloads/connector/odbc/
Select macOS operating system; download the 64-bit **DMG archive**. Run the .pkg installer inside it . (You may have to allow it to 'open anyway' under System Preferences → Privacy)

This will install the MySQL ODBC driver.

Step 3: Run the 64-bit iODBC administrator installed in 'Step 1' and select the "System" tab. Select to add a new data source. NB: Select the 'Unicode' variant of the driver, NOT the "ANSI" one.

Step 4: Give the data source a name, then set up the following parameters on it, as per:

https://dev.mysql.com/doc/connectors/en/connector-odbc-configuration-connection-parameters.html

server (the server hostname) *(this may be an IP address, but be careful if it is a dynamic IP address, then all clients will have to be reconfigured if the IP address changes)*
user (a MySQL user with permission to access the MySQL database)
database (name of actual physical MySQL database)
port (if different to default port 3306)
pwd (Optional password for username).

If the iODBC administrator application crashes (some versions appear unstable), try add one parameter at a time, OK it to save, then re-open it again.)

NB: The above username / password is NOT the Tlex/tlTerm/tlDatabase username/password.

Press the 'Test' button to test – if you get a successful connection test with the "Test" button, then you're probably 'good to go' in TLex/tlTerm/tlDatabase etc.

Step 5: Save the information and close the iODBC administrator. Run the application (e.g. Tlex) and attempt to open it. (If it's a new database that hasn't been set up yet, you'll first need to open a standalone file in TLex/tlTerm/tlDatabase and do 'Save as: ODBC' to set it up the first time. This is when you select the table 'prefix').

Important Note for 64-bit Windows Clients

*[**NOTE:** As of April 2014, you can use the "Tools / Open Windows ODBC Configuration Dialog" menu option within the application in Windows – this is the simplest way to always open the correct ODBC administrator window, rather than doing the below.]*

64-bit versions of Windows have two separate ODBC administrator panels - one for 64-bit applications, and the other for 32-bit applications. 32-bit applications (like the 32-bit versions of TLex, tlTerm and tlDatabase) can only 'see' 32-bit data sources, and 64-bit applications (like the 64-bit versions of TLex, tlTerm and tlDatabase) can only 'see' 64-bit data sources. The ODBC administrator dialog that is accessible via the Windows Control Panel on a 64-bit Windows system is for 64-bit applications (only), while the ODBC administrator dialog for 32-bit applications has

been cleverly hidden by Microsoft. For the 32-bit versions of TLex, tlTerm and tlDatabase you must use the 32-bit ODBC administrator dialog (and likewise, the 64-bit versions of these applications). The 32-bit version can be accessed by going to the Windows “Start” menu and entering:

%SYSTEMROOT%\SysWOW64\odbcad32.exe

(You can also use “Windows Key”+R to open the “Run” dialog, and entering the above.)

You can tell if your Windows system is 64-bit or 32-bit by holding in the “Windows” key and pressing “Break”, and checking the “System type”. If your Windows system is 32-bit, then you can just use the ODBC administrator dialog accessible via the Control Panel.

“Cached” ODBC (ODBC, Sped Up)

It is generally far better to use the ODBC interface labelled '**ODBC (with local cache)**' - this is like the ordinary ODBC interface, but additionally transparently keeps a local cached copy of entries that haven't changed on the local hard disk, allowing TLex/tlTerm to dramatically speed up general work and activities like searching and filtering. The first time the entire dictionary must be loaded will be slow; thereafter, it will be much faster, as it will only re-retrieve entries when they change.

Should you ever need to reset the cache, the cache folder lives in your so-called 'temp' folder, typically something like 'c:\Documents and Settings\YourUsername\Local Settings\Temp\TshwaneDJe_Cache' - just close TLex/tlTerm and delete the TshwaneDJe_Cache folder.

Entry Locking

The main difference between working on a standalone file and working on an ODBC database is the entry locking system in the latter. Entries are by default protected from casual editing, and displayed in the Preview with a padlock next to them. When you attempt to modify an entry, it will ask you if you are sure, and if you agree it will attempt to “unlock” the entry, 'checking it out' to your user. If successful, the padlock displays as unlocked and with a green tick. While you have an entry checked out, no other users will be allowed to check it out. If another user has an entry checked out, it will be displayed with a red cross through the padlock, signifying that you may not unlock it. Other users are thereby prevented from simultaneously making conflicting changes.

When you save your changes, they are committed back to the database, and the saved entries are 'checked back in', allowing other users to modify them once more.

Locking the Database

Certain types of administrative tasks, such as editing the DTD, require that nobody else be logged on to the database at that time. In order to lock all users out, select the “Dictionary/Lock database” option. If any users are connected to the database, a dialog will be displayed showing which users are logged on, while those users will also be prompted to save their work and log out. Once all users are logged out, the dictionary will be locked and you may proceed to work on it. While the dictionary is locked, nobody else may log on, and only the user who locked the database may unlock it.

Do not forget to unlock the database once you are done with the required administrative tasks.

Optimisation Tips

Working on an ODBC database may feel sluggish, particularly if the server or network is not fast. This is because entries must first be loaded from the database before they can even be displayed, and this is done dynamically and on an ongoing basis in order to ensure that you are viewing the latest data. One thing that can help a lot if the system feels slow, is to select the "**Format/Preview selection only**" option (there is also a shortcut for this in the Preview Area toolbar, which displays a "1" toggle button). Also, make sure to use the **"cached" ODBC** (mentioned above).

Notes for PostgreSQL

Timezone Settings

If using PostgreSQL, you may need to initially set up the correct timezone on the PostgreSQL server – if you are having problems accessing the database that seem otherwise inexplicable, this might be the reason.

Slow Performance (PostgreSQL)

If you experience slow entry loading performance on PostgreSQL, please try the following resolution: https://tshwanedje.com/kb/14

Commandline Options [Advanced]

The following options allow the application to be driven and controlled via the commandline.

--nouserinteraction : Disables user interaction for the session.

--runluascript : Automatically runs the specified Lua script as soon as the application has opened (and after opening any documents also specified on the commandline). You can specify multiple --runluascript items, and they will be executed in the order specified.

--autoexit : Causes the application to immediately exit after opening. If used in combination with other options, this allows the application to be automated to perform scripted commands, e.g. one could have a batch process that automatically opens some file, performs some processing (e.g. via Lua script), and then exits.

[filename] : You may optionally specify a document filename to be automatically opened. If you specify a .xml file, the built-in XML importer will be used.

Automatically Logging On to the Document Passed on the Commandline

If you are passing a document filename, you may optionally also specify a logon username and (also optionally) a password for the document. If you do, this will be used to attempt to automatically log on to the database. The syntax is as follows:

username:password?filename

Or just:

username?filename

In which case, it will still prompt for a password.

Watch Folders [Advanced]

On a per-project (per-document) basis, you can configure one or more folders in which any XML files saved to that folder, will automatically be imported into that document as long as it remains open. There are numerous settings that control the watch folder behaviour; these can be configured, and are documented in more detail, under either 'Dictionary/Properties' or 'Termbase/Properties', under the 'Properties [Advanced]' tab.

WatchFolders
Watch the given folder(s) for incoming files to be automatically imported (semi-colon separated list)

MoveImportedFilesToFolder
If specified, move successfully imported files to this folder

MoveFailedFilesToFolder
If specified, move unsuccessfully imported files to this folder

OnSuccessfulImport
0 = Move to MoveImportedFilesToFolder folder, 1 = delete file

IgnoreFilesAlreadyImported
Don't import the same file again if it's already in the MoveImportedFilesToFolder folder

ImportFileTypes
File types to automatically import (semi-colon separated list)

CheckIntervalMilliseconds
Number of milliseconds (thousandths of a second) between polls of the watch folders. Thus 5000 = five second interval.

AutoSave
Automatically save document when new data has been imported

Lua Scripting [Advanced]

TLex/tlTerm includes built-in programmability via integration of the open source Lua scripting language.

Getting Started with Lua Scripting in TLex/tlTerm or tlDatabase

Some of the main types of Lua script applications in TLex/tlTerm are:

1. **Lua Script Attributes**: A Lua script attribute is just another attribute in the document, but is special in that instead of ordinary text or numerical data, it contains a piece of Lua code that is executed in order to generate the resulting output for that attribute value. This is very similar in concept to 'formulas' in spreadsheet applications such as Microsoft Excel.
2. **Stand-alone Scripts**: Stand-alone (or external) scripts exist in the form of a text file, and are loaded and executed via the "Tools / Execute Lua Script" menu option.
3. **Smart Styles**: Using scripts to program ‘smart’ auto-punctuation or styling behaviour of fields, e.g. one may change the styling or punctuation of a field depending on whether another field appears before or after it. This may be done directly in the Styles dialog.

There are many further ways in which Lua may be used. This represents a powerful system whereby a programmer may extend various aspects of the software, or use scripting to help with data remodelling tasks.

Lua Script Attributes

A good place to start learning about Lua script attributes is to have a look at the "Lua Scripting Attribute" sample file in the Samples folder. To view the sample's script attribute contents, go to "Dictionary/Customise DTD", select the "Sense" element on the left, then select the "SenseNumLua" attribute under "Attributes of this element", click in the "Default/fixed value" box at the bottom right of the DTD editor, and press F12 to open the script in the overlay editor window. This sample demonstrates the use of a fixed Lua attribute value to generate automatic sense numbering in a manner currently not supported by the automatic numbering styles: Specifically, repeating the sense number of the parent at every subsense (e.g. "2a ... 2b ..."). The return value of the script is the value that is output in the Preview etc. (with the usual styles and so on applied, just as with any other attribute).

Note that for a Lua script attribute you can choose if the attribute value should be "fixed" or not. If "fixed" (as in the sample), the script is entered once - in the DTD editor - and the same script is always executed. If not fixed, a different script can be entered for the attribute on an individual entry/element basis.

To create a Lua DTD attribute, create an attribute as usual in the DTD editor, then select "Lua script" as the "Data type".

Note the use of the gCurrentNode global. This value always contains the node (i.e. element in the document tree) that the attribute on which the script is currently being executed belongs to. (Related to this is also a gCurrentEntry global, pointing to the owner entry.)

Note how all local variables are explicitly declared as such - in Lua, variables are global unless otherwise specified.

Stand-alone Scripts

Stand-alone (or external) scripts are basically text files containing Lua code. They can be executed via the "Tools / Execute Lua Script" menu option. By default they use a ".lua" file extension, though this is not necessary. Such a script can be used to perform pretty much any kind of manipulation on the database.

> **NB:** The text file encoding for Lua scripts *must* always be 'UTF8, without signature'.

Stand-alone Entry Scripts

"Tools / Execute Lua Entry Script" executes the selected Lua script file for every entry currently visible in the Entry List(s) of the currently active document (e.g. all that pass the filter). gCurrentEntry will point to the current entry 'on' which the script is being run. tRequestLoad() is called for you, but not tRequestModify().

You can also run Lua Entry Scripts on a subset of entries, e.g. the filtered entries, or an arbitrary tagged subset of entries. To do this, right click on the Entry List and use the "**Apply action to ...**" commands.

API Reference Documentation, Tips, Samples and Further Information

Our Lua API (Application Programming Interface) reference can be viewed via "Start menu / Programs / TLex Suite / TshwaneLua Scripting API". Good starting points are the "Detailed Description" sections of tcNode and tcDocument. (Our main Lua API reference documentation is in a file "TshwaneLua.chm", which is what this shortcut opens.)

Note that we implement Lua 5.3. This may change in future. For general Lua information (e.g. syntax or standard Lua function reference), it is suggested that you make use of the Lua reference manual, available from the http://www.lua.org/ website.

Always keep in mind the use of F12 to open the overlay editor window when editing scripts - this is virtually indispensable. It is also often useful to edit scripts in a text editor externally, and copy and paste them into TLex/tlTerm.

In our API, indexes are always zero-based unless otherwise specified. Strings are always UTF8.

> More information, plus sample scripts, are available at: http://tshwanedje.com/tshwanelua/
> Several sample files demonstrating various uses of Lua are also available in the 'Samples' project folder.

Other Uses of Lua Within TLex/tlTerm

"Smart Styles"

Lua scripts can be used to dynamically modify styles for particular fields as the output is generated, based on any possible programmable condition. See the section on "Smart Styles" in the "Styles" chapter (and the samples included with TLex) for more information.

Lua Filters

Lua scripts can be created that implement new “Filter (F5)” conditions. These take the form of .lua files, and must be placed in the 'Plugins/Filters/' folder of the application (note that an application restart may be required when copying new scripts into this folder). All .lua files listed in this folder will appear under the list of Filter conditions under “Filter (F5)”, with “Lua” and the name of the file in square brackets. An example of a Lua filter, that is included with TLex/tlTerm, is the “Lua[Modified]” filter, that filters through only entries that have unsaved changes. This is a simple script that looks as follows:

```
-- TshwaneLua Filter script for TLex or tlTerm
-- Return true if entry has modified flag set
if gCurrentEntry:HasChanged() then
      return 1;
end
return 0;
```

Lua Sorting

Custom alphabetic sorting algorithms can be implemented using the “Lua Script Sort” sort method under “Dictionary/Configure sorting” / “Add new ...”. The Lua script should implement a “main” function that takes two strings, as follows:

```
function main(s1, s2)
      -- (Return -1 if s1<s2, 1 if s1>s2, and 0 if s1=s2)
      ...
end
```

It is also possible to augment existing sorting methods in TLex/tlTerm via Lua; the Lua sort algorithm can, for example, modify the incoming strings in some way before passing them on to another configured sort method. This can also be used to implement digraph sorting.

"Click Scripts" And "Search Scripts"

Under “Dictionary/Properties”, two additional scripts can be specified for each section in a dictionary. These can optionally be used when publishing data using the TLex Electronic Dictionary Framework to customise the behaviour when users click on attributes in the Preview Area, and when users conduct a dictionary search (e.g. to implement language-specific search behaviour, such as de-inflection or morphological parsing for an agglutinative language).

User Management

Configuring User Logins

The user management system allows you to configure user logons for each user that will have access to a database. The user management system can be accessed via the “Dictionary/User management” menu option.

NB: User Management can be enabled for both standalone .tldict files and for networked client/server ODBC databases – you do not need a relational database server to use this functionality.

When you first enable user management, the software will create new “last modified by” and “created by” attributes in the DTD at entry (“Lemma”) level. Once these are present, the software will keep track of who added or modified articles. You can also manually configure attributes like this on sub-elements if you want the system to keep track of who created or modified subsections of entries (e.g. particular senses).

Notes: When logging on to TLex/tlTerm, user names are not case-sensitive, but passwords are case-sensitive.

NB: DO NOT forget your logon password(s). If you feel you might forget, write them down and store them in a safe place. Also make sure “Caps Lock” is not on when entering a password.

Deleting Users: “Delete” vs. “Purge”

If you want to remove a user from the system, by default, if you select the “Delete” command, the user's profile will not actually be removed from the system; rather, it will merely be *marked* as being deleted. Henceforth that user will no longer be able to log on. The primary reason for retaining the user profile is to be able to still display the information in the article history for where that user “last modified” or “created” articles. It also allows you to potentially undelete the user again in future.

If you select the “Purge” option, the user profile is well and truly deleted from the system. If you use this option, then for any entries created by or last modified by that user, the system will no longer know what user information to display. This can not be undone.

To summarise, the “Delete” command can thus be seen as a “soft” delete, while the “Purge” command is a “hard” delete.

It is recommended that you stick to using the “Delete” option.

Privileges System

In the user management dialog, you can select for each user what privileges they have. For example, you can untick “Export” to disallow certain users from being able to export the database to other formats, or untick “Edit DTD” to prevent certain members of the team from being able to modify

the DTD (as this is something that should only be done by someone who knows what they are doing).

Field-Specific Privileges

You may also prevent individual users from being able to edit specific attributes, using field-specific privileges. These are also configurable under “Privileges” in the user management dialog.

Monitoring and Tracking Progress

There are at least a few different ways to keep track of the progress of individual users. One is to use the “Created by” or “Last modified by” filters under “Filter (F5)” to reveal entries created or modified by a particular user or users. This could be combined with the “Search (F3)” tool to further narrow it down to particular date ranges. For example, if you filter on “Last modified by: David”, and then search for “2007-04”, it will show you all entries modified by the user David in April of 2007. This could also be taken down to the level of individual days, e.g. “2007-04-25”. See the FAQ (http://tshwanedje.com/faq.html) for more possibilities.

You can use the “**Sort by**” function under “Format (F4)” to sort the entries by the 'last modified date' (e.g. “Lemma::Modified”). This will re-index and re-sort the Lemma List according to the last modified date, showing you all the newest work at the bottom of the list. This can also optionally be combined with a user-specific filter, allowing you to see only the newest work of a particular user sorted this way.

User Progress Statistics

Another tool for tracking the progress of users is the **User Statistics** tool. This can be accessed via the “Dictionary/User statistics” menu option. This tool displays charts over time, based on the “last-modified” date of entries, of the number of entries modified on particular days – both the totals, as well as the number of entries for specific users. The date range can also be changed. A chart from the User Statistics tool is displayed in the screenshot below.

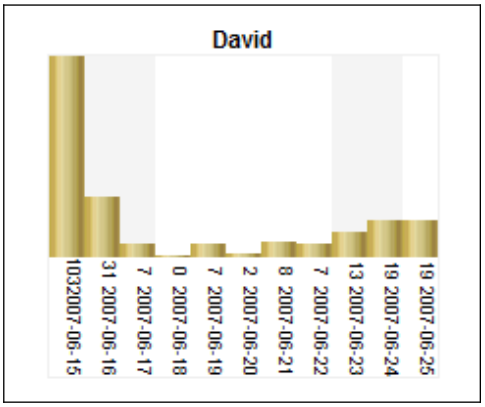


Figure 60: *A 14-day “User Statistics” chart for a single user*

Bars with a grey background denote weekends.

XSLT Export Transforms (Extensible Stylesheet Language Transformations)

The software offers various post-export “hooks” allowing you to configure automatic application of XSLT transforms (Extensible Stylesheet Language Transformations) to be applied to your exported XML data at time of export. XSLT is an industry-standard ‘language’ for applying transformations to XML data.

This could be used, in theory, to help transform from one format to another, e.g. XML to HTML, or XML to JSON.

This can also be used for tasks such as at-time-of-export data remodelling, e.g. say you have data that looks like the following being generated by the formatted XML exporter:
<element><element.type>foo</element.type>...</element>
but you need it to look like this:
<element type=”foo”>...</element>
this could potentially be done using a post-process XSLT transform.

The software has at least three different XML exporters (used for different purposes) via ‘File / Export’: All of these XML exporters allow post-export XSLT to be configured and applied.

To work with XSLT transforms, first apply the “Initial Configuration” detailed below.

Then, to set up the actual XSLT export transform, simply create a “.xslt” file named after the **current styleset** (but with .xslt extension) in the output folder where you export your XML using ‘File / Export / XML’. The default styles are normally called, “Default styles” - so if you’re still using the default styleset, your .xslt file would be called “**Default styles.xslt**”, and be located in the folder you’re exporting your XML to. The XML exporter should detect the file is present, and you should end up with both your original XML export, and the transformed file, like so:
MyExport.xml
MyExport.xml_transformed.xml

Initial Configuration

In order to apply the XSLT transforms, the software internally must run an external application called ‘XSLTProc’. You either need to first install this somewhere in your system “PATH” (on some systems like macOS it may already be present), or, in a more user-friendly way, you can install it as an ‘extension’ as follows:

1. Open your browser to https://tshwanedje.com/extensions/
2. Download xsltproc.zip
3. Run the software (either tlTerm, TLex, tlDatabase), select ‘Tools / Install extensions’ and select the xsltproc.zip file, in order to *install* the xsltproc extension into the software to use it. Restart the software. (NOTE: This only installs it locally on your own system, it does not save it with your database - so if other users on your project must also use XSLT transforms, they’ll also need to install the extension.)

You can thereafter check if the software is correctly finding the XSLTProc extension by selecting ‘Tools / Show installed extensions’, looking in the list that it shows an *xsltproc application* listed under “xsltproc”.

Multi-pass (Multi-stage) Transforms

If you need to apply several ‘passes’ of XSLT transforms to your data (with the output of each pass being used as the input for the next pass), you can do this easily by creating additional files named as “NameOfYourStyleSet(2).xslt”, “ NameOfYourStyleSet(3).xslt” and so on. The first will still be called “ NameOfYourStyleSet.xslt”, the second will have “(2)” added to the filename.

The output of all stages will be saved as separate files during the export process, so you’ll be able to see what each stage is doing - it will not overwrite the output from the previous stage.

tlTerm and TLex Online Publishing system

Introduction

TLex and tlTerm allow for 'instant online publishing' to an Online Dictionary or Online Terminology website. It does this by interfacing with an instance of "TLex Online", a publishing system(that runs on the Web server, that is licensed separately from TLex/tlTerm itself (i.e. "TLex Online" is not included with TLex/tlTerm, and requires a separate purchase). The Online Publish *exporter* itself that interfaces with "TLex Online" is, however, always included in TLex/tlTerm though, and accessible under 'File / Export' menu.

To invoke the Online Publishing command, open your database/document, and then select "File / Export / TLex Online Publishing". A settings dialog opens, which at first glance seems intimidating and contains many technical settings. However, if one follows the "Best Practices" for configuration and workflow, it is generally (once configured *correctly*) relatively easy to use, and can offer close to a 'one-click style' publishing workflow for editors. See the section below '"Best Practice" Configuration and Workflow'.

[Advanced] Technical Details

The "TLex Online" system is built using PHP, a popular scripting language for developing websites, that runs on the web server. It is a set of PHP scripts, effectively a PHP framework, typically customised for each user's requirements. It typically requires some amount of customisation by a programmer to initially set up your Online Dictionary or Online Terminology website that uses it. (The amount of customisation work required can thus differ vastly depending on the extent and complexity of individual requirements.)

"Best Practice" Configuration and Workflow

It's important to note that the settings you choose in the "File / Export / TLex Online Publishing" dialog itself are saved only *locally on your computer*, as local user settings, and not inside the TLex or tlTerm file or database. Thus, if you set up all the settings correctly here, but then open the file for publishing on a different computer, your settings would not be there. (If you're not careful, this could lead to problems such as duplicate entries on the server, e.g. if different settings are inadvertently used for e.g. TitleID or EntryIDBase.)

To **prevent** this, and to make it more user-friendly and 'quick and easy' for an editor to publish (close to 'one click' style publishing), the **'correct' workflow** is to **first** set up an 'online publishing profile' that is saved into the TLex or tlTerm file itself (or ODBC database).

This is done as follows:

1. Open your TLex file (or tlTerm file, or ODBC database)
2. Select the "Dictionary/Properties" menu (or "Termbase / Properties" if a terminology database)
3. Select "Properties [Advanced]" tab
4. Double-click the "OnlinePublishing" "**NumProfiles**" setting, and change it **from 0 to 1** (if it's not 0, someone already did this!)
5. Click "OK"
6. Save the database, **close it, and re-open it**.

7. Select the “Dictionary/Properties” menu **again**, and select “Properties [Advanced]” **again**
8. You should now see many more “OnlinePublishing” settings. Configure these accordingly (as per below), and then ‘Save’ the document again.

These settings that you configure under “Dictionary/Properties” (or Termbase/Properties) are saved centrally *in the TLex file itself* (or tlTerm file, or ODBC database), and thus become the sort of ‘master settings’ for publishing for that dictionary or termbase.

When you invoke the “File / Export / TLex Online Publishing” command now, TLex (or tlTerm) automatically **fills in** the settings in the export/publish dialog (though you can still override if need be), with the settings stored in the ‘Dictionary/Properties’ section. This allows you to do close to ‘one click’ style publishing *if* the settings are correctly configured, but *also* allows an advanced user to change the settings at the ‘last minute’ if needed in the actual export dialog (for example, to change the PublishURL to publish to a test/staging/development version of the Online Dictionary or Online Termbase instead during development).

The most important settings are:

PublishURL: This is the web server address (URL) of where your “TLex Online” instance lives, and receives the ‘publishing commands’ when you do a publish. This typically looks something like:
https://yourdictionary.com/receive.php
(If you’re unsure, ask your website programmer, or TshwaneDJe, what this setting should be.)
PublishURLPassword: A special secret custom password that is configured by your website programmer on the server side to help prevent someone malicious from accessing the server at that PublishURL. (This may be left blank in ‘Dictionary/Properties), in which case it will ask you for the password each time you invoke the publish command - this is more secure than storing it in Dictionary/Properties, but is more inconvenient. For best security it is recommended to thus leave this field blank and let the user fill it in each time.)
TablePrefix: The exporter generates its output data in a set of MySQL tables. This option allows you to have multiple TLex or tlTerm databases/titles published in parallel to the same server, by prefixing the TablePrefix to the table names to generate a unique, different set of tables names. For example, you might want to use “l” for your live database and “s” for a separate staging database for testing. This all depends on your particular requirements. If you’re not sure what to set it to, and only have one database and one title, then just select something simple, like “l”, and stick to it everywhere. (Note that this setting also needs to be configured once-off in the “TLex Online” PHP configuration files by the website developer, or by TshwaneDJe if we are setting things up for you.)

Other settings you typically may need to configure are:

TitleID: A single “TLex Online” instance can hold many dictionary and/or terminology titles (and even a mix of dictionary and terminology titles within the same database). If you are publishing multiple ‘titles’ online, then it is important to set the TitleID to some unique value that you choose to correspond to each title, e.g. 1, or 2, or 3 for each title.
EntryIDBase: As a rule of thumb, this typically should be set this to the TitleID multiplied by 10,000,000.

If you only have one dictionary (or one termbase) being published, then the above two can just be left on their defaults.

Other Optional Settings:

StopWords: This may be set to a comma-separated list of 'common words' *not* to be added to the search index (unless it's an actual "important" headword (lemmasign) in a dictionary, or term in a termbase - the 'important' ones will be added always). This can help prevent the user from being overwhelmed by thousands of "useless" results if they search for something common like "the". So for example, you might set this to e.g.:

the,a,is,has,of

NB: If you have an actual dictionary entry "the" (or terminology entry for "the"), TLex (and tlTerm) are smart enough to 'know' that it should be added *anyway* even though it's on the stopword list.

OverridePathImages: This may be important if your dictionary or termbase has image or sound files, in which case you may need to set this to the server-side path (as a URL) containing your image files. So e.g. if you have, say, "horse.jpg" attached to your "horse" entry, and that file lives on your server at:
https://mydictionary.com/media/horse.jpg
then you may set the 'OverridePath' to e.g. "/media/" or alternatively (in some cases, e.g. if the entries may be viewed on third party sites or Apps via the Web search API backend, it may be necessary to use the full URL), "https://mydictionary.com/media/"

BatchSize Setting and Publish Speed

If the publishing speed seems slow, try raise the BatchSize setting to 1000000. This can especially help for large dictionaries or terminology lists.

However, *if* (and only if) you get *timeout errors* from the server during publishing, then set it back to e.g. 50,000, or experiment with values in this region, e.g. 40,000.

The default BatchSize is 50,000. The setting breaks the publishing job up into 'many smaller tasks', so a *lower* BatchSize causes a *slower* publishing speed (as it has to send many more smaller requests to the server during the publish process), while a *larger* BatchSize means a faster publishing speed.

[Advanced] The reason the slower setting is unfortunately the default has to do with how PHP works: By default, many PHP servers have a server-side timeout setting (typically e.g. 30 seconds). Unfortunately this setting can usually only be controlled from the server side by a server administrator – so in some cases you might not be able to change this. This PHP timeout setting can thus be annoying for our publish purposes. (If your administrator has access to the PHP settings, you could also ask them to disable the PHP script execution timeouts, and then you should be able to safely use a high BatchSize such as 1000000 which will then speed up publishing.) TLex/tlTerm cannot override the server timeout setting – this is the reason the BatchSize setting exists.

[Advanced] Generating SQL Command Files without actual publish

[Advanced/Technical] The "File / Export / TLex Online Publishing" system can be used to generate SQL commands for constructing a potential MySQL online dictionary database for housing the published dictionary data. This can be done by simply leaving the PublishURL setting empty. The publisher/exporter will send the output commands to the OutputFolder setting. These are basically

just plaintext SQL, and can be combined (concatenated) to produce what is effectively a MySQL dump file (e.g. “cat *.txt >> dump.sql” in that folder). Your programmer may be able to use this to import the database into their own Online Dictionary or Online Terminology setup, if you are not using the TshwaneDJe “TLex Online” system.

[Advanced] Working with Multiple Online Publishing Profiles

The following is of relevance typically if working with the “many dictionaries from one database” functionality.

In almost all cases, the “NumProfiles” setting in “Dictionary/Properties” (or “Termbase/Properties”) should be “1”. However, in some cases, you may have more profiles configured here, e.g. 2 or 3. This would usually be if you are using the TLex “Many dictionaries from one database” functionality.

For example, if you have one dictionary document from which you generate a “Full” and a “Pocket” ‘view’ (i.e. version/output), and you have two StyleSets (“Full” and “Pocket”), then the “**StyleSet**” setting in Dictionary/Properties should be set to “Full” for the first, and “Pocket” for the second. TLex then ‘smartly’ selects which to publish based on your selected StyleSet in TLex or tlTerm itself. This also smartly allows each of these two dictionaries to get separate, different title IDs and EntryID ranges, or even different PublishURLs.

[Advanced] Publishing to Your Own Custom Online Platform

If you have programmers who have developed (or are developing) your own online dictionary or online terminology site, then (depending on your requirements), we can help you set up a publishing workflow from TLex or tlTerm customised for your own online platform - contact TshwaneDJe to discuss your requirements.

Troubleshooting

Duplicate Entries

If you are ending up with duplicate entries after a publish, this is usually the result of having used the wrong ‘TitleID’ or ‘EntryIDBase’ settings during a publish. Subsequent publishes in that case end up publishing a second copy of your title into a different title ID and entry ID range ‘slot’. If you have only title, this can be fixed easily typically by setting the “OutputCreateTables” and “OutputDropTables” settings to ON in the ‘File / Export / TLex Online Publishing’ dialog. **WARNING**: If you **do** have several other titles in the same database and TablePrefix, setting these to ON will wipe them out, and you’ll have to republish those too.

[Programmers] MySQL Publish Database Schema

The following information is generally of interest for programmers only, and presents a brief overview of the publish-database tables on the "TLex Online" MySQL server:

"i" Main search index

"f" Full-text search index
"r" TLex smart-references (entries linked to other entries, by their generated entry ID)

"l" Entry list (may or may not be used, depending on your requirements)

"ii" Reverse index (for bilingual dictionary) (may or may not be used, depending on your requirements)

"j" [New] Similar to main search index but allows field-specific information, to allow for functionality like 'search within definitions only' for end users

[Advanced/Programmers] Notes on the meaning of the "scores"

The 'scores' in the main search index are used to distinguish between more "important" search keywords (e.g. main lemma signs in a dictionary, or actual terms in a termbase). The score value is a percentage, with values of 100 meaning 'most important' (i.e. most relevant as a match for the dictionary end-user). So for example, for the *actual* dictionary entry "dog", the search index would get an entry for "dog" with a score of "100", along with the entry ID for the "dog" entry. But if the word in question appears only somewhere within the dictionary text (e.g. somewhere within a usage example or definition of an entry other than "dog"), then "dog" will be added to the search index but with a *lower score*, i.e. less than 100.

This allows programmers to distinguish 'main entries' (e.g. the actual 'dog' entry) vs potentially relevant (but probably less important for end-users) entries that merely contain 'dog', if the user searches for that. **IMPORTANT**: The default "TLex Online" search functionality *already handles this* - but if you have your own programmer working with the database directly to implement your own search functionality, the programmer will need to know this.

TLex has a system that allows it to 'understand' that compound phrases (e.g. "draw the line") should also be '100% matches'.

[Advanced/Programmers] Notes on Entry IDs

Note that although entry IDs in TLex and tlTerm have their own unique IDs (within the internal XML 'document tree'), when you do a publish, for technical reasons, new (*different*) unique IDs are generated for the *online* versions of the entries (starting at your EntryIDBase setting). There is currently no specific mapping back from the generated IDs to the original TLex or tlTerm entry IDs, though the information on the original entry ID might be added in future to the MySQL output.

tlCorpus (Time-saving) Linked Mode

"Linked Mode" is a time-saving feature designed to potentially significantly speed up some normally repetitive lexicography and terminology tasks, and other language work.

When running in "Linked Mode", the tlCorpus application and TLex (or tlTerm) are linked to one another and can "talk to each other", allowing them to offer additional *integrated functionality*.

Quickly Create New Entries: In Linked Mode you can right-click on any word in the tlCorpus search results window (*NB: requires the tlCorpus 'search results colours' option to be on*), and quickly **add a new entry** in TLex/tlTerm for the right-clicked word/term.

Adding TLex Usage Examples: When right-clicking on a line in the tlCorpus search results, you can select "[Linked Mode] **Auto-Add Example Sentence**" to automatically **send** the sentence in the right-clicked corpus line to TLex to be automatically added as a lexicographic usage example to the current entry in TLex. (Note you may have to first select the desired target sense in the TLex Tree View.) This feature can greatly speed up the process of adding corpus-based usage examples to a dictionary.

(Note that without Linked Mode, one can still "send" tlCorpus results to TLex to be added as usage examples, but there's an additional step: Right-click on search result line, select "Copy Example Sentence for TLex", select the desired target 'Sense' in the Tree View in TLex, then press Ctrl+Shift+F7 to add it.)

Instantly Launch Corpus Searches: Linked mode works **both ways**: While working in TLex or tlTerm, you can e.g. right-click on a term or headword in the Entry List or parts of the Tree View (e.g. translations, terms), and under the '**Text commands for ...**' menu, select "**Search in tlCorpus [Linked Mode]**" to instantly launch a tlCorpus search for the right-clicked word/term. This makes it quick and easy to launch tlCorpus searches while doing lexicography, terminology or translation work. (This is similar to existing functionality in the 'Corpus F6' tab, but for the full tlCorpus application rather than the small F6 window.)

When Linked Mode is active, the tlCorpus title bar should say "[**Linked Mode Enabled**]" - if not, Linked Mode is *not* active, and you should try again to launch it correctly linked. See below for instructions and troubleshooting.

This informal video shows tlCorpus Linked Mode in action:
https://youtu.be/FLhI9Ut0pAk?t=25

Starting tlCorpus in Linked Mode

To start tlCorpus in Linked Mode, first run TLex (or tlTerm), then on the Start Page, under "Tasks/New/Open", next to "Launch tlCorpus" click on "**Launch Linked**". This should run tlCorpus for you with Linked Mode *active*. *(Note that TLex Suite includes TLex, tlTerm and tlCorpus, but tlCorpus is not included with tlTerm Standalone and may be a separate purchase.)*

Alternatively, you can also first run TLex (or tlTerm), then under the "Tools" menu, select "Launch tlCorpus (linked)".

Note: You may get a message like the below in Windows regarding firewall access when running in Linked Mode – select to 'Allow access' as per the screenshot below. This allows the applications to establish their internal link and talk to one another.

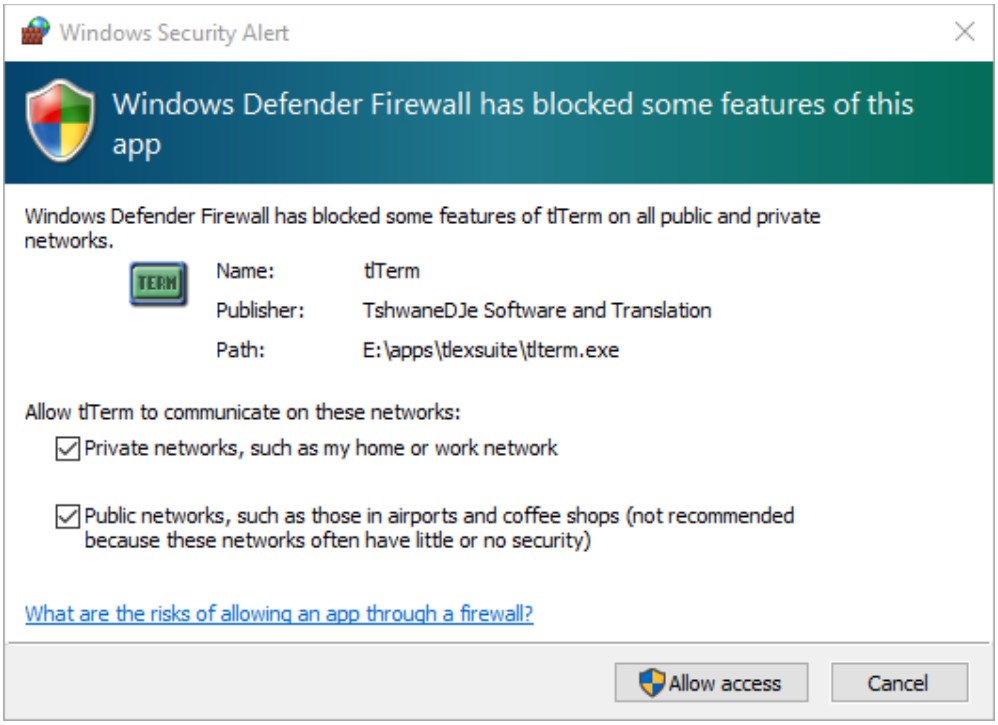


Figure 61: You may get the following warning from Windows when running "Linked Mode" - select 'Allow access' as shown in the screenshot.

Troubleshooting Linked Mode

If "Auto-Add Example Sentence" doesn't work, try first select the desired target 'Sense' in TLex in the Tree View.

[Advanced] Note that if you are using your own entirely custom DTD (i.e. not the default TLex DTD), then this might also not work unless you set up your DTD "Special types" correctly in the DTD editor, as TLex relies on these 'special types' (configured for you automatically only in the *default* TLex DTD) to "know" e.g. which field is a usage example, which is a sense, and so on.

If all Linked Mode features aren't working, it may be that Linked Mode is not correctly active. Look for the text "[**Linked-Mode Enabled**]" in the title bar of **tlCorpus** – this will appear if tlCorpus was correctly run in Linked Mode.

Note also that if you close the linked TLex/tlTerm while tlCorpus is running, then the Linked Mode 'connection' will be broken, and you may have to close and re-run both (with Linked Mode active) to start again with Linked Mode active.

[Programmers] Linked Mode "Web API"

Technically, the linked mode functionality uses (effectively) a sort of internal ‘Web API’. This makes it theoretically possible for other applications to potentially also send or receive ‘commands’ and interact with these applications.

For example, it should be possible for third party corpus applications to be configured by a programmer to be able to ‘send usage examples to TLex’. Or, other applications could potentially be programmed to be able to send text strings to tlCorpus to have it launch searches.

Contact TshwaneDJe if you may have a use case for this and are interested in setting something like this up.

Multi-section Documents

Introduction

A document such as a monolingual dictionary usually has one “section”, and a bilingual dictionary usually (but not always) has two “sections” (which in e.g. TLex we may work with using side-by-side view). A terminology database typically has one section, though it’s possible to add more (e.g. see the multi-section terminology sample).

The software allows more sections to be added, and this can be useful in some cases for certain projects or for certain types of documents. E.g. in a more complex book-style document, you can have a section for each chapter. There can be dozens of sections in such a document.

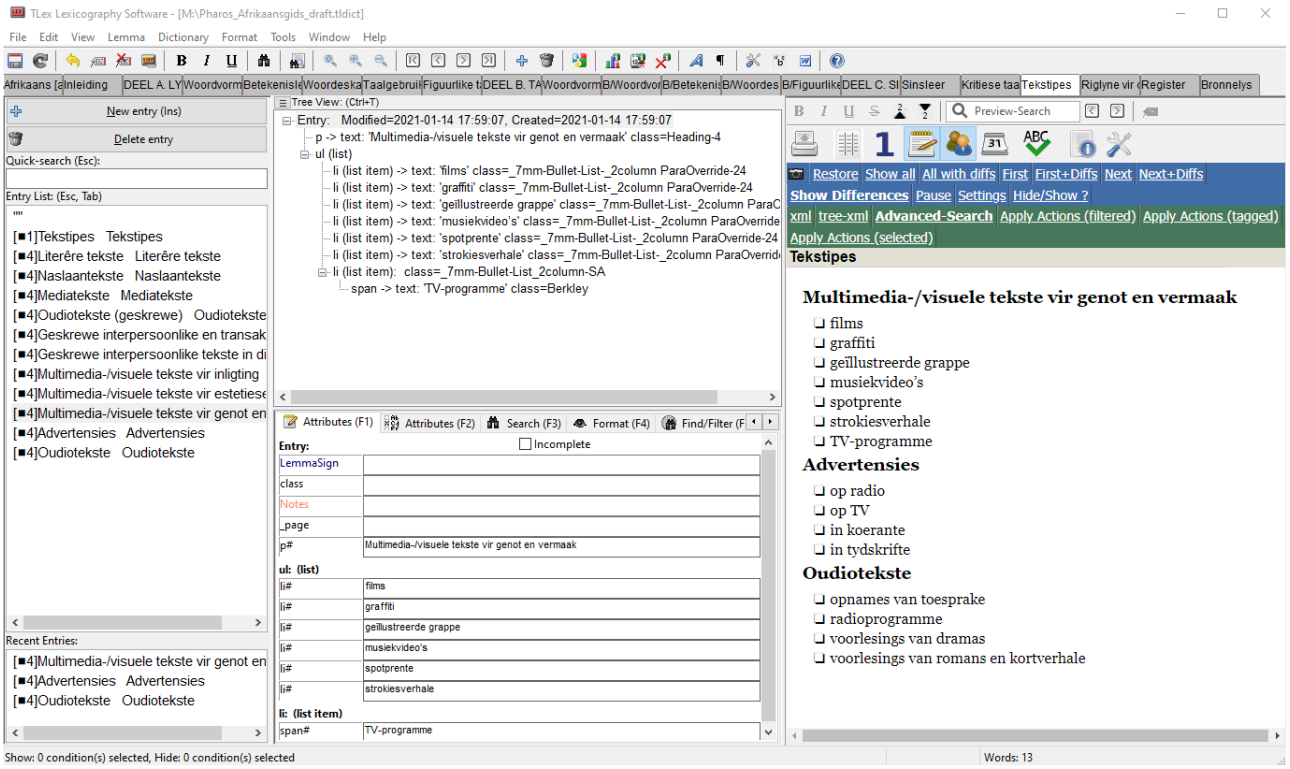


Figure 62: Sample 'book-style' multi-section document (with tabbed user interface) with each chapter in its own section

You can click on a tab to activate and view that section. You can also use Ctrl+F6 to switch to the next tab, or Ctrl+Shift+F6 to switch to the previous tab. Under the “Window” menu, you can also use the “First window” and “Last window” commands to help navigate to different sections.

[Intermediate] Adding, Renaming or Deleting Sections

You can add, delete or rename sections using the “**Edit sections**” tool under the “**Tools**” menu.

> ⚠ **Warning:** If you *delete* a section, and then save, the data in it will be *gone*! Make backups before trying this.

In *TLex* specifically, if changing from a monolingual to a bilingual dictionary, you can use the ‘Dictionary/Properties’ dialog box to add a section by changing to “bilingual” type.

OpenAI Integration (GPT) [v2023+ version only]

Feature Availability: v2023+ Version only

NB: Please note the new OpenAI integration functionality is only available in the new v2023+ release of TLex, tlTerm etc. – if you have an older license, this specific new functionality may not be available unless you upgrade to a new release.

Using OpenAI with TLex (and/or tlTerm, tlCorpus, tlDatabase etc.)

If you want to use the OpenAI (GPT/AI) integration in TLex Lexicography Software (or tlTerm, tlCorpus or tlDatabase), you'll need to obtain a secret 'API key' from OpenAI.

GPT stands for "Generative Pre-trained Transformer", and is an AI (Artificial Intelligence) Language Model that is designed to generate human-like text responses to queries, and can do many tasks that can help augment and speed up lexicographic work.

The 'API key' is a unique text string identifier linked to your account, that you generate in the OpenAI website when logged in (see instructions below), that you keep secret to yourself, that allows you access to OpenAI's functionality from within software like TLex. It is used to authenticate you to OpenAI.

You enter the key into your TLex settings, and TLex then "talks to" the OpenAI servers, sending your 'API key' with each request in order to authenticate you, and authenticate your access to their services.

Please note OpenAI is a third-party service and they are not affiliated with TshwaneDJe Software, and we have no responsibility for the content their service generates. It is important to first test and make sure it suits your needs.

If used properly, it can significantly increase productivity, as it can help automate tasks like creating basic starter dictionary entries, sample definitions and usage examples etc., as well as help with tasks like checking grammar, or even creating some content for a small starter corpus or to supplement a corpus.

NB: Also note how much you get *out* of the service also depends how much you put in – e.g. if you put in some extra effort at the start to develop decent custom user prompts in TLex or tlTerm that are more well-**customized** *for your project*, you'll get more useful and productive results than with just the mere more generic built-in integration (for example, by customizing the prompts, you can fine-tune for your desired languages you work with, desired structure and DTD, desired style and length of content, and so on. TLex (and tlTerm, tlCorpus etc.) have the ability to configure your own 'custom user prompts' for OpenAI in the application Options/Preferences.

(The quality of the content may vary widely especially for less-frequently-spoken languages, as the GPT AI service is a Language Model, that has been trained on texts available on the Internet, so languages for which there is very little data on the Internet, it has not been well-trained on, and therefore is more likely to generate incorrect information for less-frequently-spoken languages.

Nonetheless it is a potentially very useful tool for virtually any potential project, if used properly, and understanding its limitations.)

Also note though that (for advanced users, it is possible to 'fine-tune' their language model with additional training texts for particular tasks, though that is outside the scope of this User Guide.

NB: You should take note of the Terms of Service of the API, and of any 'rate limits' – see below.

Also note that due to the way these Language Models work, they can and do generate incorrect information, so the output must be checked. These aspects are likely to improve in coming years.

Technically, it's a large-scale neural network that has been trained on many texts for generating human-like text and responding to queries, but due to the way they work, the information is not always correct – it depends heavily on factors like available 'training data' for it when it is being 'trained'.

NOTE: "GPT" vs "ChatGPT": Note that (as of writing ~Mar 2023) there are two similar but different APIs: The commands labelled "GPT" in TLex are the general GPT-3 API (which ChatGPT is *based on*), and the ones labelled "ChatGPT" are specifically the ChatGPT API (which is one particular customization of the same GPT-3 'engine'); which should you use? It's suggested to try each one, and see which you get better results from (though in general there shouldn't be a major difference).

How to get an OpenAI key?

To get an OpenAI key, you will need an OpenAI account on their website (https://openai.com/). Once you have created your account, you can follow these steps to obtain your API key:

1. Open the API page (https://platform.openai.com/) using a Web browser
2. **Log in** to your OpenAI account if you already have an account. If not, click "**Sign up**"
3. In the OpenAI website (platform.openai.com), **click on your profile icon** (typically near top right)
4. Select '**View API keys**'
5. **Generate a new secret API key**. Copy it somewhere safe, and do not give it to anyone. (It is possible to later delete keys or generate new keys, but you **can't** re-retrieve keys again if you *lost* a key.) This key must go into the settings in your TLex etc., see below.

How to enter your OpenAI key in TLex Lexicography Software?

Once you have your OpenAI key, you can enter it into TLex (or tlTerm, tlCorpus etc.) by following these steps:

1. **Open TLex Lexicography Software (or tlTerm, tlCorpus, tlDatabase etc.)**
2. Click on the "**Tools**" menu and select "**Options**" (or App menu / "**Preferences**" on Mac).
3. Find the "**OpenAI**" settings, and double-click the "**APIkey**" setting
4. Paste (or enter) your OpenAI API key generated above
5. Click "OK" or "Close"
6. After entering your OpenAI key, you should be able to use the OpenAI integration within TLex Lexicography Software.

Note: This key setting is **NOT stored in your actual project files/data/database -** the key is only stored locally on that computer as a local user setting, and you may have to repeat this process for tlCorpus and tlTerm etc. if needed on the same or other computers, as each application has its own set of settings.

If you have different users, they may need separate/different OpenAI accounts (and thus keys).

Using the Service in TLex, tlTerm, tlCorpus etc.

In many parts of TLex, tlTerm, tlCorpus and tlDatabase there are general "Text Commands" menus when you right-click on things (e.g. right-clicking on an an entry in the Tree View or Entry List) - see the separate chapters on 'Text Commands' menus.

In the latest releases that support OpenAI integration, these Text Commands menus have been extended to include access to many new OpenAI GPT query commands, e.g. from 'Explain', to 'Check grammar', 'Create list of senses', to outright writing (sample) entries.

In general, you should *check* the quality of what it generates, fix any problems (or generate more new e.g. entries if the entry was not good enough), extend the entry with additional information as needed etc.

In tlCorpus right-clicking on any word in the search results ('colour' results option) also provides access to GPT commands like 'Explain'.

OpenAI TLex/tlTerm/tlCorpus Settings

See "Tools/Options" (or on macOS "Preferences") under the "OpenAI" for many other settings, e.g. such as auto- node options etc.

Automated al

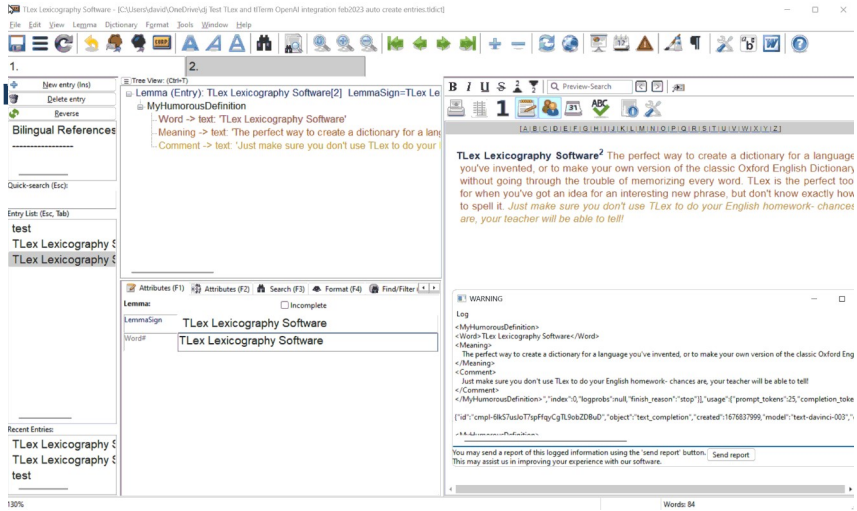


Figure 63: Entry for "TLex Lexicography Software" automatically created by GPT using OpenAI integration in TLex

Figure 63: Automatically created entry by GPT using TLex OpenAI integration

You'll notice some of the right-click commands for OpenAI are labelled "XML" - in general, if you want it to auto-create entries in TLex, use the "XML"-labelled menu commands, and enable entry auto-creation in the application settings.

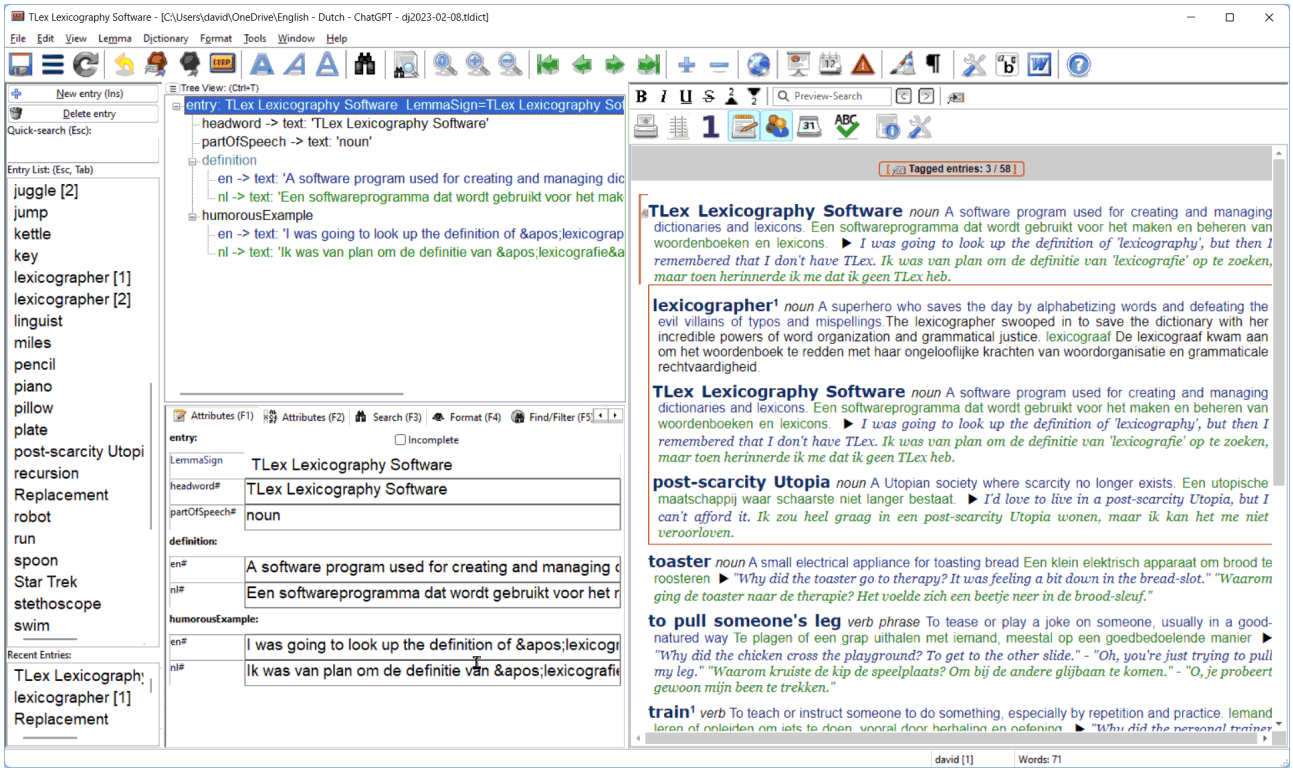


Figure 64: Automatically created entries by GPT using TLex OpenAI integration. See https://dictionaryq.com/GPT-dictionary/

See https://dictionaryq.com/GPT-dictionary/ to view the above 'Mini English – Dutch Humor Dictionary' online, created by GPT with TLex's OpenAI integration.

Word and Excel Integration with AI integration

Note that on Windows, this functionality is also available in the Word pop-up, for any text (as well as the Excel integration pop-up window)

This includes multi-line, e.g. can be used to create multiple entries at once, so you can use Word etc. to feed it with automated batch queries.

Click on the bold part in the Word popup window to get all the same OpenAI GPT integration (including your own custom prompts - see the section on custom prompts).

Multi-Document Search and AI integration

Note also that, like the Word integration window, the same OpenAI integration commands are also available in the Multi-Document (MDM) search dialog. So e.g. if you have, say, XLIFF and/or TMX sources loaded in the MDM tool while you're working, you can also use the OpenAI commands – e.g. if you search a TMX or XLIFF source, for example, see a word or phrase in the results you want the AI to auto-create an entry for you, it should work to do so via clicking in the MDM window (e.g. in search results) to open the menu commands.

Custom User Prompts

There are a many useful default built-in GPT prompts, but to get much more out of the service, it's a good idea to also create your own custom prompts.

In "Tools/Options" on Windows (or via "Preferences" menu on macOS), it is possible to add your own custom user prompts. These will then automatically appear on your right-click menu along with the built-in GPT commands.

Note that are only saved locally to your computer, they aren't saved in your project file(s).

The custom prompts are a multi-line list where each line contains a custom prompt, and each line is tab-separated, with the text before the tab being the label that will appear on the menu, and after the tab the text that will be fed to GPT, e.g.:

My custom prompt<tab>Write a long English – French dictionary entry, with definitions, synonyms and antonyms, and translations in French, for the word; return the entry in structured XML: $term$

You can add as many custom prompts as you want, and feel free to add many to experiment to 'fine-tune' the usefulness of the results you get back.

"$term" will be replaced by whatever you right-click on when actually using it.

Query Rate Limits and Timing

Note that you should not 'spam' the server with many thousands of queries at once, or you may violate OpenAI's Terms of Service. They also do enforce "rate limits" (not necessarily a Terms of Service violation, but an 'enforced temporary slow-down'). The rate limits are relatively generous, but may depend on which plan you have with them and other factors. In TLex/tlTerm etc.'s "batch mode", by default it waits around 5 seconds between queries – this should be a reasonably safe default in most cases, but if get rate limit errors, try increase the setting for delay between queries and try again.

For more information though (and possibly more up to date information), check here:
https://platform.openai.com/docs/guides/rate-limits

Setting Up LM Studio or ollama (litellm) AI Servers

Feature Availability: v2023+ (v15+) Version only

Setting Up LM Studio AI or litellm ollama AI Servers in TLex/tlTerm/tlDatabase

This section of the user guide provides instructions for configuring LM Studio AI servers within the TLex software (or tlTerm, or tlDatabase), enhancing the application's capabilities with artificial intelligence integration.

Step 1: Accessing AI Server Settings

To begin, open the software and go to the "Tools/Options" menu (or in Mac, the application Preferences). In this dialog, you will find a new option labelled "AI/Servers." This is the area where you will manage your connections to LM Studio AI servers (or ollama servers).

Step 2: Configuring AI Servers

Within the AI/Servers section, you can add the addresses of your active LM Studio (or litellm ollama) servers. If your server is running on the same computer, enter one of the following addresses (default port for LM Studio is 1234; default port for 'litellm --model ollama/modelname' is 8000):

```
# LM Studio localhost
127.0.0.1:1234
# ollama litellm localhost
127.0.0.1:8000
```

If the server is on a different machine in your Local Area Network (LAN), input the LAN IP address of that machine, for example:

```
10.0.0.10:1234
```

Step 3: Utilizing AI in TLex/tlTerm/tlDatabase

After configuring your AI servers, the TLex interface will include new functionality. When you right-click in the application, the AI context menu will now display additional options for each configured AI server, alongside the current existing OpenAI options. This allows for easy access to multiple AI servers and their respective tools directly through the TLex interface.

Custom prompts also work, and your OpenAI custom prompts will also be available to these AI servers.

Ollama setup

To install ollama models, install 'ollama' and then use the following to download custom models:

```
ollama pull mistral
```

To run them as AI servers, do:

```
pip install litellm
```

Then you can use the following commands for downloaded ollama models to run your own server (default port is typically 8000):

```
litellm --model ollama/mistral
```

Note that it should work with ollama custom Modelfile instances too, so you can for example use your own custom ollama Model that is tailored for particular projects, languages, data structures, and your available hardware etc. to run these.

LM Studio Setup

You can download and install the LM Studio software, then use its user interface to download AI models. You can then use the integrated server section to run a server (default port 1234).