The GNU Emacs TNSDL Mode Manual

for version pre-2.0.1

Peter Tury

This manual is for GNU Emacs' tnsdl-mode what is made to help programming in the TNSDL programming language.

Table of Contents

1 Preface

If reading off-Emacs (e.g. on printed paper or a '`.pdf`' or '`.html`' file(s)) I recommend reading only Section 2.1 [Set up GNU Emacs], page 2 and Section 2.3 [Set up tnsdl-mode], page 5: after the installations, reading this manual inside Emacs is much more convenient (e.g. all the crossreferences will work).

If you're "just interesting" and don't want to install anything, then I suggest reading only Chapter 3 [Everyday Usage], page 7.

If you are an experienced Emacs user, then Chapter 3 [Everyday Usage], page 7: it is probably more than enough for you. If you have time for it, Section 2.2.3 [Getting Help], page 4 and Section 2.2.4 [Useful Commands], page 4 also: you might find unknown useful Emacs utilities. If you haven't installed Info files previously, then Section 2.3.2 [Documentation], page 5 for instructions. Also Appendix A [Supported Features], page 13.

2 Getting Started

This tnsdl-mode is written for every TNSDL users. Some of them might be unfamiliar with GNU Emacs. In this chapter we summarize everything for a successful start.

For other aspects See ⟨undefined⟩ [Top], page ⟨undefined⟩.

For help about TNSDL itself, go to the TNSDL nwiki.

2.1 Set up GNU Emacs

GNU Emacs is part of DMXSee. However, only an extremely old, outdated version, what is not supported by tnsdl-mode from version 2.0.0 onwards. Thus you have to install another, newer Emacs for yourself. Here follows how.

For general info see `xxx`.

For MS Windows installers, see `xxx`.

For MS Windows I recommend using the official precompiled binaries from `xxx`. Their “installation” means unzipping the downloaded file and manually setting some things in MS Windows:

1. Set `HOME` environment variable to point e.g. ‘`C:\USERS\`’. Your ‘`.emacs`’ file will be here. Remember its location! See ⟨undefined⟩ [Windows HOME], page ⟨undefined⟩.
2. Associate ‘`.sdl`’, ‘`.sdt`’, ‘`.spd`’, ‘`.sst`’ etc. files to Emacs.

Also See ⟨undefined⟩ [Help installing Emacs], page ⟨undefined⟩, and ⟨undefined⟩ [Using an already running Emacs process], page ⟨undefined⟩.

For info about starting and stopping Emacs See ⟨undefined⟩ [Entering Emacs], page ⟨undefined⟩, and ⟨undefined⟩ [Exiting], page ⟨undefined⟩.

For full details about start options See ⟨undefined⟩ [Emacs Invocation], page ⟨undefined⟩.

2.2 GNU Emacs Lifebelt

2.2.1 Emacs Key Notation

Some keyboard notation used in Emacs manuals, helps, etc.:

Key	**Emacs notion**
ALT	M
SHIFT	S
CTRL	C
DOWN ARROW	DOWN
F1	F1
DELETE	DEL
ENTER	RET
SPACE	SPC
BACKSPACE	BS
PGUP	PAGEUP

Examples:

Key sequence in Emacs notion	**Meaning**
C-SPC	Hold down CTRL and push SPACE (set mark)
M-W	Hold down ALT and push W (copy)
C-W	Hold down CTRL and push W (cut)
C-Y	Hold down CTRL and push Y (paste)
C-M-K	Hold down both CTRL and ALT and push K (cut a balanced expression)
S-DOWN-MOUSE-1	Hold down SHIFT and click with the left mouse button (mouse-set-font)
C-U 10 C-X $	Hold down CTRL and push U; then type *1 0*; then hold down CTRL and push X; then type $ (hides all lines indented by 10+ space)

For more info See ⟨undefined⟩ [Basic Keys], page ⟨undefined⟩, ⟨undefined⟩ [User Input], page ⟨undefined⟩, and ⟨undefined⟩ [M-x], page ⟨undefined⟩...

2.2.2 Emacs Terminology

These are not "official" definitions, just short scratches for beginners to help understanding further Emacs documentation.

point	The **cursor** and/or its location (in terms of character position). See ⟨undefined⟩ [Point], page ⟨undefined⟩.
kill	Item is copied and deleted. Well known name for this functionality: "**cut**".
yank	Previously "killed" or "copied" item is **paste**d.
frame	It is a "**window**" from the windowing system's (e.g. MS Windows) point of view.
buffer	It is a piece of text. Usually a whole file's content.
window	A part of a "frame". One frame can consist of one or more windows. Windows can be splitted into more (smaller) windows.
visit a file	Opening it in Emacs.
fontification	Syntax highlighting.
minibuffer	The (generally) one-line area at the bottom of a frame, where you can e.g. type commands in. Note: Emacs communicates with you mainly via the minibuffer! See ⟨undefined⟩ [Minibuffer], page ⟨undefined⟩.
command	An Emacs function designed to be callable by end users. See ⟨undefined⟩ [Commands], page ⟨undefined⟩.

For many more terms explained more exactly See ⟨undefined⟩ [Glossary], page ⟨undefined⟩. For learning more about the screen See ⟨undefined⟩ [Screen], page ⟨undefined⟩.

2.2.3 Getting Help

Most of the “interactive” help within Emacs is available in Info. To learn about its usage See ⟨undefined⟩ [Top], page ⟨undefined⟩.

Type C-H ? anytime for a list of available help possibilities. Some of them:

C-H B	List all key sequences usable at the moment (in the current window!).
C-H I	Opens the Info reader with enourmos amount of available local, hypertexted documentation.
C-H K	Type a key sequence or click a mouse button (e.g. on a menu): its help is shown.
C-H L	Helps finding out what triggered the last events: show your last 100 keystroke.

See ⟨undefined⟩ [On-line manual], page ⟨undefined⟩.

See ⟨undefined⟩ [Learning how to do something], page ⟨undefined⟩. Note especially ‘`refcard`’ in it!

2.2.4 Useful Commands

Several basic Emacs key combinations and commands are mentioned above (See Section 2.2.1 [Emacs Key Notation], page 2, and Section 2.2.3 [Getting Help], page 4).

Here I list some more, what are especially useful for TNSDL development. (tnsdl-mode’s own commands are not listed here. For them See Chapter 3 [Everyday Usage], page 7.)

If you read this in Emacs’ Info reader (recommended), then move the cursor (point) onto the function names below one by one, and press C-H F RET on them: C-H F calls the Emacs function (command) `describe-function` with default value derived from text found around point. (Note: Emacs communicates with you mainly via the minibuffer. Check it!) Then RET tells that the default’s descripton is looked for. The “description” contains the function’s name, the actual key combination(s) bound to it, a short description about its usage, etc. The best is to try them immediately in a trial (TNSDL) buffer...

`indent-for-tab-command`

`isearch-forward` – `isearch-backward`
`isearch-forward-regexp` – `isearch-backward-regexp`

`dabbrev-expand`

`query-replace` – `query-replace-regexp`
`replace-string` – `replace-regexp`

`comment-dwim`

`just-one-space` – `delete-horizontal-space`
`delete-blank-lines`
`kill-line` – `kill-whole-line` – `kill-sexp` –> ⟨undefined⟩ [Erasing], page ⟨undefined⟩

`find-tag`

`forward-sexp` – `backward-sexp`

`mark-sexp`

`split-window-vertically` – `split-window-horizontally`
`other-window`   `delete-other-windows`

`kill-buffer`

`make-frame-command`

`count-lines-region` –> ⟨undefined⟩ [Position Info], page ⟨undefined⟩

`goto-line` –> ⟨undefined⟩ [Moving Point], page ⟨undefined⟩

`upcase-word` – `downcase-word`

`undo`

`hs-minor-mode`

`color-theme-select` (requires ColorThemes)

For others See ⟨undefined⟩ [Help], page ⟨undefined⟩. ⟨undefined⟩ [Basic], page ⟨undefined⟩

Do not forget to check refcard.ps in the '`emacs/etc`' directory.

2.3 Set up tnsdl-mode

Everything what is needed for tnsdl-mode is downloadable from `http://xxx`.

In this section we describe all the details about how to use these files: how to "install" the functionality and the documentation. Source files also described with full details.

2.3.1 Functionality

tnsdl-mode's functionality is implemented in one single file: '`tnsdl.el`'. Since Emacs can interpret source files, you should have either this source file, or its byte compiled version '`tnsdl.elc`' for using tnsdl-mode.

To make Emacs find this file when needed, copy it into a directory where Emacs looks for such files: type *`C-h v load-path RET`*. This brings up a help window showing the value (and meaning) of the variable `load-path`. Any directory in the list will suffice; '`emacs/site-lisp`' is dedicated to such purposes, so you can choose it safely.

After copying '`tnsdl.el(c)`' into such a directory, Emacs can find it. Now you have to order Emacs to load it when it starts. For this, locate your '`.emacs`' file (see Section 2.1 [Set up GNU Emacs], page 2) and add this line to its end: `(load "tnsdl")`. Go to the end of the expression you've just typed (so place the cursor (point) immediately after the closing paren `)`) and type *`C-x C-e`*. From now on all tnsdl-mode functionalities should be available. Try it by opening (visiting) a tnsdl file.

See ⟨undefined⟩ [Lisp Libraries], page ⟨undefined⟩.

2.3.2 Documentation

tnsdl-mode's documentation comes in the following forms:

Info	for online reading (inside Emacs or in another info reader),
PDF	for printing and
HTML	for browsing in a web browser.

You know how to handle HTML and PDF files.

The Info file '`tnsdl`' can be "installed" for Emacs, so you will reach it via the standard Emacs help system. Here is a step-by-step instruction how to install.

1. Locate documentation coming with Emacs. You should find it under '`your Emacs documentation directory/info`'. If you don't know where is it, try searching e.g. for file '`elisp-8`'.
2. Copy the info file '`tnsdl`' into this directory.
3. Edit the file '`dir`' in this directory: add a line like

   ```
   * TNSDL: (tnsdl).        Emacs mode for developing TNSDL PRBs.
   ```

   after e.g. the line

   ```
   * Info: (info).          How to use the documentation browsing system.
   ```

   Next time you type *`C-h i`*, the hypertexted root of tnsdl-mode's documentation will be linked in: type *`RET`* on it and start browsing!

2.3.3 Sources

Here is a full list of tnsdl-mode's source files along with their usage. If you want to contribute to tnsdl-mode development, use (enhance) these files. Otherwise you can safely skip this subsection.

'`tnsdl.el`'
: Contains the Emacs Lisp implementation of all the functionality of tnsdl-mode. Byte compiled '`tnsdl.elc`' file can be generated from it.

'`tnsdl.texinfo`'
: Source of all the documentation: both Info file '`tnsdl`', '`tnsdl.pdf`' and HTML documentation files are generated from it.

'`test_tnsdlmode.sdl`'
: A (partly pathological) TNSDL file to test syntax highlighting, indenatation, move commands, etc.

'`test_tnsdlmode.el`'
: "Unit tests" for tnsdl-mode development.

3 Everyday Usage

The purpose of tnsdl-mode is to help everyday sw development work.

Most of tnsdl-mode is nothing more than supporting and customizing of standard Emacs functionalities.

Source code typing, modification, reviewing and navigation are helped as well as compilation and other related activities. Details follows in this chapter.

3.1 Syntax Highlighting

This is standard Emacs functionality, but I added a menu for it: Tnsdl `>` Font-lock.

Fontification can be totally switched off or switched on.

You can choose from a lot of ready-to-use color schemes or you can define your own fonts to use. I recommend the former one using ColorThemes: http:\\xxx.

When fontificaton is switched on, several levels can be applied:

Level-0
: highlights only the most basic elements: strings, comments, compiler directives and STATEs, PROCEDUREs and TYPEs.

Level-1
: adds fontification for all keywords

Level-2
: additionally highlights type names (like ‘`byte`’, ‘`bool`’, etc. or ending with ‘`_t`’), constant names (ending with ‘`_c`’ or ‘`_ec`’), variable and parameter names (starting with ‘`g_`’, ‘`l_`’ or ‘`p_`’), etc.

For more, general info See ⟨undefined⟩ [Levels of Font Lock], page ⟨undefined⟩.

3.2 Indentation

tnsdl-mode indents lines automatically when *RET* is typed: both the just-closed and the just-started lines are indented.

Typing *TAB* once tries to indent intelligently: indentation happens (instead of inserting spaces or tabs) if point wasn’t in a place where inserting whitespace seems to be reasonable.

Second and following *TAB* typings do xxx. This in practice can be used for alignement quite comfortably.

Regions etc. can be reindented using the Tnsdl menu.

Each line’s indentation is calculated from patterns found in the actual and preceding lines. When the actual line is empty (=just started), calculation can be wrong easily. In such cases type the line and hope *RET* at the end of the line will reindent well.

When automatic identation does a wrong job, indent the line manually, and hope following lines’ automatic indentation will be calculated in a good way from it.

3.3 Navigation

A lot of standard (and sometimes most convenient!) possibilities are not mentioned below. Check the *Edit* menu (especially its *Search* and *Go To* submenus) for them. I use *Incremental Search* most of the time.

Also note that ⟨undefined⟩ [Top], page ⟨undefined⟩ is supported. Try it: *M-x speedbar RET*.

3.3.1 Nearby

Use standard Emacs commands: ‘forward-word’/‘backward-word’, ‘forward-sentence’/‘backward-sentence’ (this is quite handy!), etc.

3.3.2 Far

Skipping bigger entities (e.g. branches of DECISIONs or any code enclosed by <keyword>-END<keyword> pairs) is supported by the ‘forward-sexp’/‘backward-sexp’ functions.

When you want to “skip out” from such a block, then ‘tnsdl-goto-block-start’/‘tnsdl-goto-block-end’ can be used. These are in the Tnsdl menu, but no keys are bound to them by default.

Skipping to the start/end of the actual PROCEDURE has its own menu and keys. ‘beginning-of-defun’/‘end-of-defun’ do these.

Many other (probably interactively less useful) tnsdl-specific move commands are defined: check (via *C-h* `a`) the commands starting with ‘`tnsdl-goto-`’.

Note: It is a good practice to use ‘point-to-register’ to store any loaction where to you want to go back later.

3.3.3 Away

Skipping to a specific PROCEDURE or TYPE is easiest using TAG files. Generate/refresh the TAG file using the menu *Etag*. Sometimes you have to load the freshly generated file(s) using ‘visit-tags-table’. Then use ‘find-tag’. Usually it defaults to the right name, but you can type any name you want.

3.4 Hide and Show

Some bigger entities (e.g. branches of DECISIONs or complete PROCEDUREs or their long FPAR or DCL parts or long comments, etc.) can be hidden temporarily. This functionality is usually referred elsewhere as “folding”. Use ‘hs-minor-mode’ for it. (It is not activated by tnsdl-mode by default, you have to switch this minor mode on by yourself: type *M-x hs-minor-mode RET*. Then its menu appears...)

3.5 Compilation

Currently I suggest compilation is the following way.

1. Create your own MS Windows batch file what compiles from command line. E.g. a file ‘`buildmyprb.cmd`’ could look like this:
   ```
   call meenv 5.1-0
   build -dstdout -mk2cd -rm2cd -test -timestamp my__scmx.pac.
   ```
   Note: these two lines are enough for compilation.
2. Copy this ‘`buildmyprb.cmd`’ into the directory where your ‘`my__scmx.pac`’ resides.
3. Open a file from this directory in Emacs. E.g. it is a good practice to visit the ‘`my__scmx.pac`’ file and set the ‘`.img`’’s id in it.
4. ‘`M-x compile` *RET* `buildmyprb.cmd` *RET*’
   (Note: after the first time, you can use ‘`M-x recompile` *RET*’ instead.) After this, compilation starts (you may have to open an SPM connection before it). Compilation

messages appears in their own buffer. tnsdl-mode adds knowledge to highlight error and warning messages differently and makes them links.

5. Click on an error/warning message and the file at the problematic line will be opened. If Emacs ask you to locate the directory where the problematic file resides, then usually the best is to open any TNSDL file from that directory and check-in TNSDL menu's Compilation > In search path before the compilation process. After this, all source files in that directory fill be found automatically by Emacs.

4 Customization

To customize tnsdl-mode use the menu *`Tnsdl > Customize`*.

It is usually a good idea to bind your favourite commands to key combinations, so you don't have to type *`M-x my-fav-command RET`* always. You can choose any key combination you like (even "self inserting" keys like TAB or RET or simple symbols like ~ or letters etc. can be rebound), but *`C-c LETTER`* are reserved for such purposes (=for users): no major or minor modes should use them.

To bind e.g. *`C-c b`* to the command 'tnsdl-goto-block-start', put this line into your '`.emacs`' file (append to its end):

```
(define-key tnsdl-mode-map "\C-cb" 'tnsdl-goto-block-start)
```

5 Contribute

If you simply use this tnsdl-mode and send your comments to the author, then you help a lot! (For names ⟨undefined⟩ [xxx], page ⟨undefined⟩.)

However, you are free to modify any source files (either documentation or test or implementation sources). If you do it, please inform the author, so I can add you modification to the “official” version.

If you comply with my coding conventions, then I can easily merge your enhancements into the code. Here I list some of them.

5.0.1 Coding conventions

- Each function’s, option’s, ... name starts with ‘`tnsdl-`’.
- Everything (functions, variables, ...) is documented in its documentation string.
- Functions’ (intentional) returned values should be mentioned explicitly in their docstring.
- Whenever it is possible, add functionality via supporting official minor modes and features instead of reinventing the wheel. Examples: hs-minor-mode, imenu, ‘forward-sexp’, etc.

6 Copying

This manual is for GNU Emacs' tnsdl-mode what is made to help programming in the TNSDL programming language.

Appendix A Supported Features

Here is a (hopefully full) list of standard Emacs features what are supported by tnsdl-mode. I.e. these can be used (in addition to the special tnsdl-mode-features) “out of the box”.

Type *`C-h f RET`* on them to get more info.

- which-func-mode
- imenu (makes TAG files unnecessary locally)
- hs-minor-mode (“folding”)
- font-lock-mode (syntax highlighting)
- compilation-mode (to skip to the problematic source code line from the compilation log)

Appendix B News

- v2

  This is a complete rewrite from scratch. Made by Peter Tury. It works only with Emacs v22 or later.

  New features:

  - complete documentation
  - standard Emacs customizability (via `M-x customize-mode`)
  - multi-level and more exact syntax highlighting
  - folding
  - tnsdl-specific code browsing (‘forward-sexp’, etc.)
  - enhanced(?) automatic indentation
  - TAG support
  - compilation support

  Removed features:

  - abbreviations (I found ‘dabbrev-expand’ enough)
  - skeletons (I didn’t use them)
  - unknown functionalities (probably there were some)

- v1

  First(?) tnsdl-mode was created and maintained by Harri Menp until 1992. Later Toni Arte and Tero Venetjoki maintained it for many years. Its variants can be used for older Emacses (from version 17(?)).

  It knew:

  - one-level syntax highlighting
  - automatic indentation
  - menu
  - imenu support
  - abbreviations
  - skeletons

Index

(Index is nonexistent)