WaitZar User's Guide

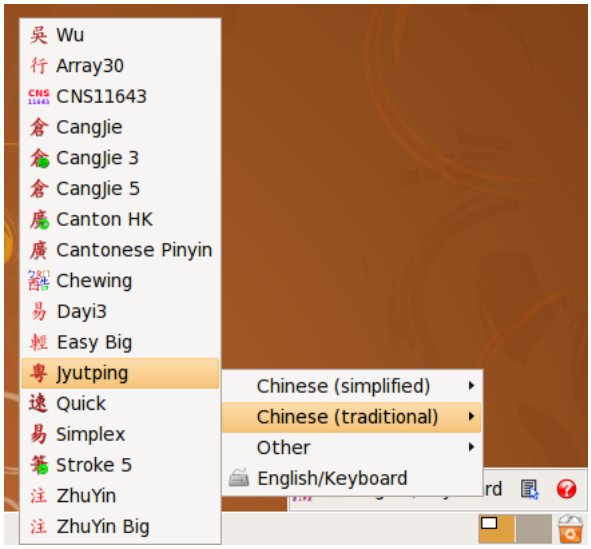
SCIM Wait Zar version 0.0.1

Table of Contents

1. Introduction to SCIM and WaitZar

If you're a Linux user who knows how to write a complex script like Chinese or Thai, chances are you've used SCIM before. Even some European users prefer SCIM, although their languages do not require such a complex solution. The **scim-waitzar** project extends SCIM to support the Myanmar language.

Figure 1. A screenshot of SCIM running on Ubuntu, showing a wide variety of Chinese input method engines



WaitZar is a romanisation of Burmese designed specifically for fast typing on a standard keyboard. Given the fluid nature of the Myanmar unicode specification (e.g., the 4.0, 5.0, and 5.1 encodings are mutually incompatible) and the wide variety of *ad hoc* encodings (i.e., Zawgyi-One partial, Win Innwa's ASCII kludging, etc.), WaitZar provides a degree of solidity and flexibility above and beyond that provided by most other input schemes for Burmese.

The official project page of **scim-waitzar** is:

http://scim-waitzar.googlecode.com/

If you would like to keep casually up-to-date on WaitZar's Windows and Linux projects, you might consider joining out mailing list, which averages about one announcement per month:

http://groups.google.com/group/waitzar

2. Requirements & Installation

WaitZar will run on almost any Linux distribution that also supports SCIM. In fact, by installing the **scim** package (or installing SCIM from source), you should already have everything you need. WaitZar is recommended for Ubuntu Linux, but it has also been tested on Debian Linux. Please inform us if **scim-waitzar** works (or breaks!) on your favorite brand of Linux.

Unlike the Windows version, WaitZar under Linux requires a significant amount of installation effort. The most stable method is to build **scim-waitzar** from source; see Section 2.3. However, a great many build steps are simplified if your system supports Debian packages. See Section 2.1 if you're used to using apt-get, and Section 2.2 if you'd prefer to (manually) use dpkg. Finally, Section 2.4 explains several very important things to consider after installation (using any of the three methods.) Until WaitZar gets fully integrated into Ubuntu's *Language* manager, you will have to consider these.

You should have at least one of the following Unicode 5.1-enabled fonts installed on your system. Please see Section 2.4 for more information on how to install fonts.

- Padauk
- Parabaik
- Myanmar 3

You should install one of these fonts (for example, **ttf-sil-padauk**) in order to view this document properly.

Note Regarding Fonts

Traditionally, Burmese fonts have defined their own *ad hoc* encodings, which has caused a lot of confusion among novice computer users and experts alike. Even previous versions of the Unicode encoding (for example, 4.0 and 5.0) had non-negligible errors and incompatibilities. Fortunately, Unicode 5.1 has resolved most of these, and the three fonts listed all render this document with no errors.

Developers, please note that, when discussing character encoding incompatibilities (in, say, the "Wait Zar User Interface Specification") the documentation will use the Zawgyi-One font & encoding, due to its unambiguous representation of variant forms.

2.1. Installation Using Apt

The recommended way of installing WaitZar is to use **apt-get** or the Syanptic Package Manager. In order to do this, you will have to add a line to the sources file. To do this, open a console and type:

```
gksudo gedit /etc/apt/sources.lst
```

...and add the following repository lines at the very end. Note that these links are temporary. After we finish alpha testing, we'll put the **scim-waitzar** package at a more suitable location.

```
deb http://www.comp.nus.edu.sg/~sethhetu/scim_wz_0.0.1/ ./
deb-src http://www.comp.nus.edu.sg/~sethhetu/scim_wz_0.0.1/ ./
```

...then, save the file and close **gedit**, and type:

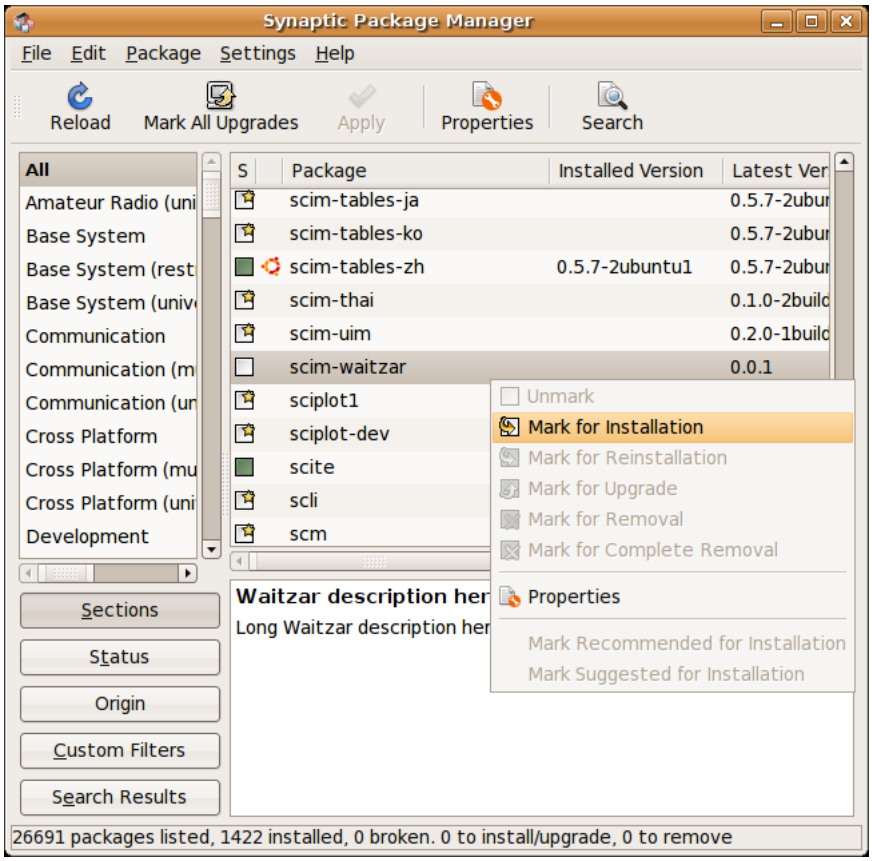
```
sudo apt-get update
```

This will update the list of packages. Finally, install **scim-waitzar**:

```
sudo apt-get install scim-waitzar
```

Of course, you can also use Synaptic at this point:

Figure 2. After reloading your repositories, you can simply install WaitZar like any other package.



2.2. Installing the .deb File Directly

In general, the apt repository will never go offline. Nevertheless, you can always download and install the .deb package manually. Grab the latest version from:

http://code.google.com/p/scim-waitzar/downloads/list

...and then run the standard **dpkg** install command:

```
sudo dpkg -i scim-waitzar_*.deb
```

...from the directory you downloaded the package to. You must then restart X (log out, log in). In case you are not familiar with **dpkg**, note that un-installation can be done from any directory, and requires only the logical name of the package:

```
sudo dpkg -r scim-waitzar
```

2.3. Building & Installing from Source

Building from source is actually quite simple, thanks to the GNU autotools. Here are the steps for the Ubuntu operating system, although they should work with any **automake**-compatible system. These instructions are for the "developer hat", since users will probably just use **apt-get** to install WaitZar. If you are wearing your "user hat", you should still be able to follow along.

1. Install necessary packages, preferably with **apt-get**:

   ```
   sudo apt-get install subversion autoconf automake gettext libtool scim-dev g++
   ```

2. Make a directory for the source files, and download them with **svn**:

```
mkdir ~/scim-waitzar
svn checkout http://scim-waitzar.googlecode.com/svn/trunk/ ~/scim-waitzar
cd ~/scim-waitzar
```

3. Reconfigure the system:

```
autoreconf
```

4. Run through the standard autotools lifecycle:

```
./configure
make
sudo make install
```

5. Log out, then log in again. WaitZar is now (almost) ready to use.

2.4. Fonts & Configuration Extras

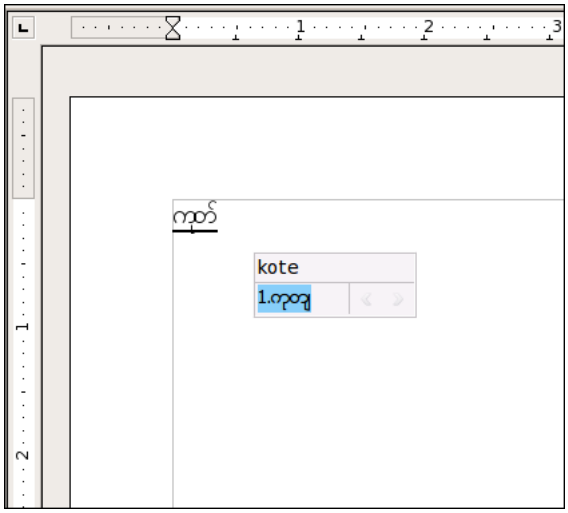
Although most Linux programs have been properly internationalized, a great many users simply do not trust "tacking another layer" onto input (that layer being SCIM). Hence, simply installing WaitZar will not be enough —you'll also have to enable it, play around with **fontconfig**, and probably install some fonts, too. Here are some helpful hints and tricks. It should go without saying that, although these techniques are harmless, we take no responsibility for any unintentional side effects they have on your system.

Enable Chinese — Making every application recognize SCIM is not easy; there are reports that "SCIM won't work in Open Office" which are only half true. Fortunately, there's a cheap trick that works well in Ubuntu: just enable Chinese, and the system will figure all this out for you. Click on "System" then "Administration", then "Language Support", and check the box marked "Chinese". Several package files will be installed. Finally, check the box that reads "Enable support to enter complex characters", and click "Ok". You will probably have to restart. Note that, if you do not have at least one complex language installed (like Chinese), checking the box will have no effect. That is why we recommend enabling Chinese.

Figure 3. The "Language Support" window. Make sure the "complex characters" checkbox is checked.

Finagle FontConfig — SCIM does not dictate which font should be used for its main interface, relying on *fontconfig* like every other program does. That means the wrong font can be picked for the wrong encoding. For example, here is what WaitZar looks like if Zawgyi-One is chosen by mistake.

Figure 4. Trying to type “kote” with a mis-configured font selector. Note that this will not actually affect the typed word, just the list of choices.



We recommend listing at least one Unicode 5.1 font in local.conf's “prefer” tags. We've created a sample file for you that also makes your text sharper, and has Chinese fonts listed properly. You can get it here:

http://scim-waitzar.googlecode.com/svn/trunk/local.conf

...copy it to

```
/etc/fonts/local.conf
```

...using, for example:

```
cd /etc/fonts && sudo cp local.conf  local.bak
sudo wget http://scim-waitzar.googlecode.com/svn/trunk/local.conf
```

You'll probably have to restart X. (Log out/Log in).

Install Some Fonts — You'll need at least one font installed to use WaitZar effectively. Some fonts have packages to make installation easier; for example, the *Padauk* font has **ttf-sil-padauk**. If no package exists, you have several options. The simplest is to make a user-level fonts directory:

```
mkdir ~/.fonts
```

...and then copy any fonts you download to that folder. You should not have to restart X, but you might need to restart any running application you want to use the font in.

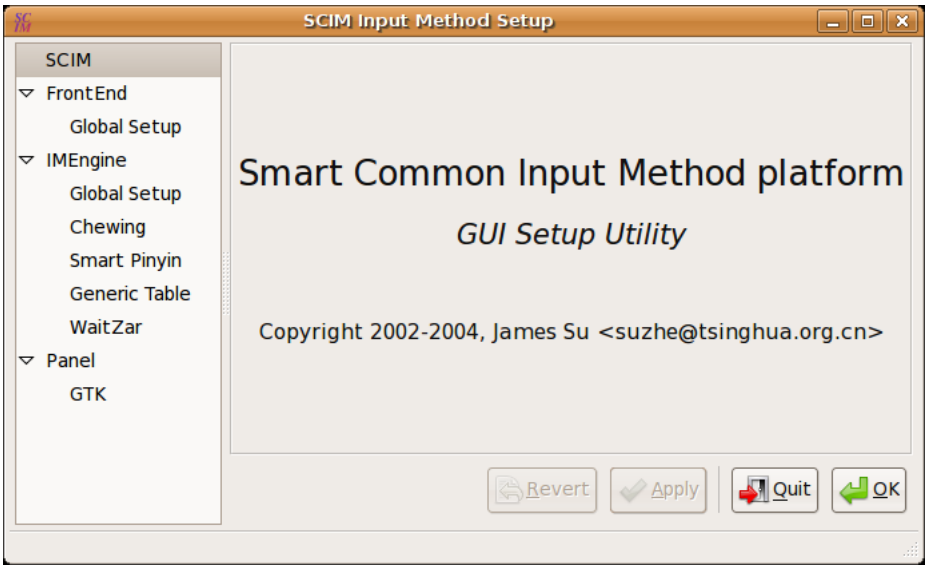
3. Basic Usage

3.1. Introduction to SCIM

After you install **scim-waitzar**, you should take some time to get used to SCIM's unique feel. You will see a small keyboard-shaped icon in your tray, right-click it, and choose “SCIM Setup”. This panel can be quite confusing for new users, so we'll briefly cover it here.

The “Front End → Global Setup” panel contains SCIM's hotkeys, and various global settings. You will probably want to change the hotkey from Ctrl+Space to *anything* else, if you are a programmer.

Figure 5. The SCIM setup window.

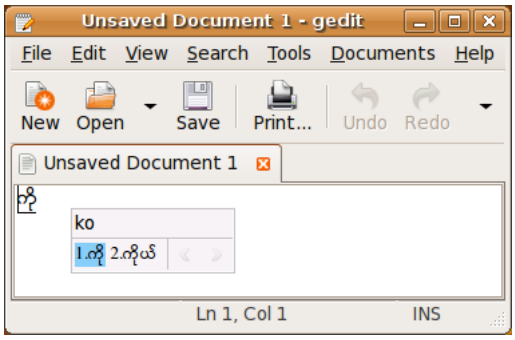


The next section (IMEngine) has settings for each “Input Method Engine”. Recalling that SCIM provides support for multiple languages, it makes sense that each language should have its own customizations. The final section, “Panel → GTK”, holds most of the visual settings for SCIM. I recommend changing the “Show” option (under “Toolbar”) to “Always”, at least until you get used to how SCIM behaves. Note that when you click “Ok” to exit the setup panel, SCIM will warn you that you might need to restart SCIM for any changes to take place. The sure-fire way to do this is to log out and then log in again. Optionally, you might right-click on the SCIM icon and choose “Reload Configuration”. In some cases, the operating system will reload SCIM for you. Regardless, you won't be playing around with the WaitZar settings that much anyway.

3.2. Typing Your First Myanmar Sentence

After configuring SCIM just the way you like it, start **gedit** and then look at the lower-right corner of your screen. You'll see the SCIM toolbar; move the mouse over it and it will expand to show that you are typing in Englihsh. Click on the language and choose “Burmese” to switch to WaitZar. (Note that if this panel is hidden, you can always click on the SCIM icon in your tray to switch to Burmese.) Then, type “ko”. You will see two possibilities presented to you. You will also see the word “ကစဉ့်” inside of **gedit**. This is one of the nice features of WaitZar on Linux: you can keep your eyes on the document you are typing, and only have to look at the list of choices when something doesn't match.

Figure 6. Choosing between two "ko"s.



Hit “space”, and the cursor will move right; the list of choices will disappear. Now, type “2”, and you will see the Burmese numeral “၂”. Finally, hit “Enter”, and the words you typed will no longer be underlined. Why did this happen? Basically, WaitZar “pre-edits” anything you type. Hitting left and right works correctly in WaitZar because it delays direct typing of words. Although this may take some getting used to, it is actually quite natural, and there are several shortcuts to save you time.

3.3. Additional WaitZar Hotkeys

By default, SCIM lets you switch to the *last* language you used by hitting Ctrl+Space. Hitting it again will switch to English.

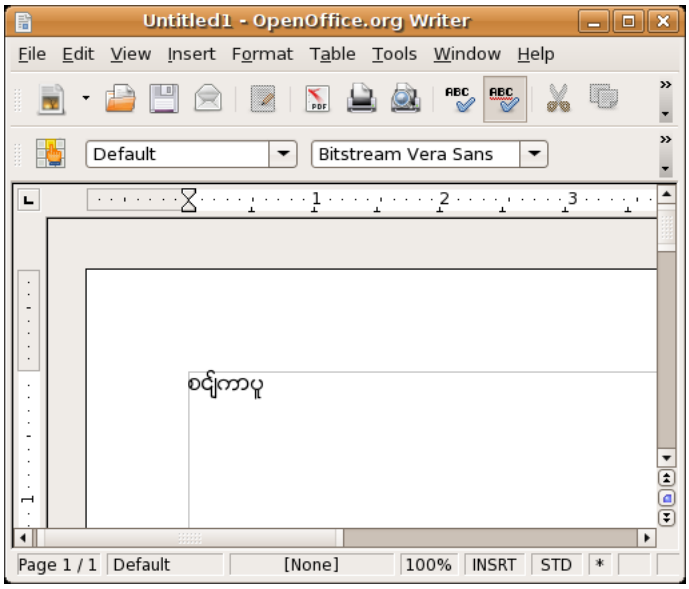
When you are typing in WaitZar, several other hotkeys also take effect:

- Pressing any letter will type that letter, and attempt to form a word.
- Pressing left and right will move the cursor. When you are typing a word, left and right switch between the various choices.
- Pressing “enter” will confirm the typed sentence. If you are typing a word, hitting “enter” is the same as hitting “space”.
- Pressing “space” will move the cursor right. If the cursor is all the way to the right, hitting “space” will confirm the typed sentence. When you are typing a word, hitting “space” will choose the currently-selected choice.
- Hitting the number keys (0-9) will type the burmese numeral (၀-၉). If you are typing a word, hitting a number will “shortcut” to that word. (Try typing “ko2” to see this work).
- Hitting the period or comma keys will type the Burmese full or half stop characters (“၊” and “။”). If you have typed part of a sentence, this will also confirm the sentence.
- Hitting “esc” will cancel the sentence or word you are typing.
- Hitting “backspace” and “delete” will function as expected, deleting letters or words depending on the context.
- In the rare case that you are using an absolutely tiny display, hitting up and down will scroll the list of choices if there are too many to fit on the screen at once.

3.4. The Open Office Glitch

There are several glitches related to WaitZar; most are covered in Section 5. However, one particularly nasty bit occurs in Open Office Writer –to see it for yourself, change the font (in Open Office) to Padauk, type “singapore”, and then hit “space” twice to finalize it. The result is a mess of glyphs:

Figure 7. Open Office seems to like resetting the font for you, even when this is actually incorrect.



Fortunately, this is easy to fix: just select all the text and change the font to “Padauk” again. For some reason, Open Office reverts the font to *Bitstream Vera Sans*, an otherwise-excellent font that, unfortunately, cannot entirely handle Burmese just yet.

Even more fortunately, this only has to be done once: the first time you create a new document. Even if you delete everything on the page, Open Office can still manage to change the font back to Padauk the second time you enter Burmese text.

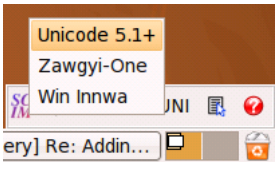
On a side note, this glitch is far worse on Windows, which doesn't provide a reasonable catch-all font. In this case, the first Burmese word you type in Open Office will actually be invisible!

3.5. Changing the Output Encoding

A great many websites use encodings that conflict with the established standard. Criticizing them will do no good; these *ad hoc* encodings were all developed to suit a particular niche, and they will likely be around until the end of time.

WaitZar supports two of these encodings: the Zawgyi-One font encoding, and the Win Innwa font family encoding. (Win Kalaw, for instance, is in the Win Innwa font family). To change the encoding, start typing in Burmese, and then click the “UNI” icon in the SCIM toolbar. Now choose the encoding you want. This will reset the next time you log in; if you want to change the encoding used by default, see Section 3.6.

Figure 8. Changing the output encoding.

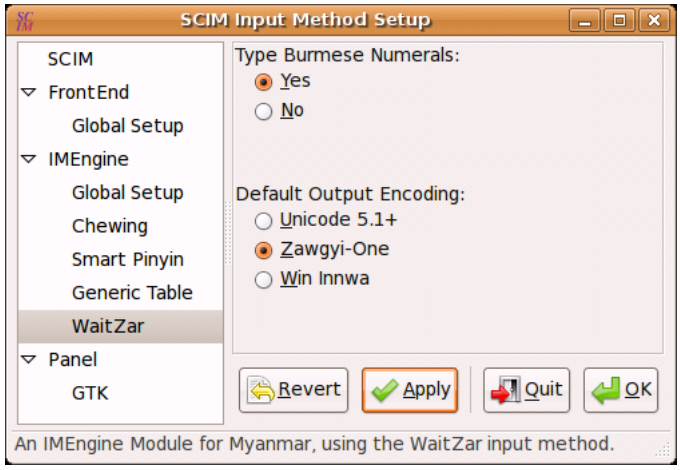


3.6. A Note About config.txt

For Windows users of WaitZar, it should come as a mildly unpleasant surprise that config.txt does nothing. The proper way to change WaitZar's configuration settings is through the SCIM settings panel, on the “WaitZar” tab.

You can currently only change two settings: how Burmese numbers are typed, and the default output encoding. A screenshot of this panel follows.

Figure 9. WaitZar's settings: here, the user has chosen to type numbers in Burmese, and to start WaitZar with Zawgyi-One by default.



3.7. Adding custom words to mywords.txt

WaitZar's dictionary is drawn from online forums, books, and news articles, and it contains most of the words needed by most of the users. However, it is possible that you may find it lacking a word you need. Or, you might type a word so often (like your name) that you want to shorten it. For this, you will add lines to the file "mywords.txt" of the form:

myanmar = roman

For example:

ဗ = ma

Note that, for legacy reasons, you will have to enter the word you require in the Zawgyi-One encoding. If you have the word in a different encoding, the following web site provides excellent software for converting it:

http://burglish.googlepages.com/fontconv.htm/

The only remaining detail is *which* mywords.txt file to edit. For global changes to WaitZar, you can change /usr/share/scim-waitzar/mywords.txt. For user-level changes, you should change ~/.scim-waitzar/mywords.txt. The latter way is preferred. You should use the UTF-8 encoding for this file; for example:

```
mkdir -p ~/.scim-waitzar
gedit --encoding UTF-8 ~/.scim-waitzar/mywords.txt
```

4. Finding a Word

This section describes how to find a word. It is under development...

4.1. The WaitZar Romanisation

This section describes the WaitZar romanisation. It is under development.

4.1.1. Default Romanisation

This section describes how to find most everday words. It is under development.

4.1.2. Special Romanisations

This section describes how pat-sint and foreign words are romanised. It is under development.

4.1.3. Shortcuts

This section describes shortcuts like "aung" and the like. It is under development.

4.2. Dictionary Lookup

This section tells where to go for a dictionary lookup of words. It is under development.

5. Known Bugs

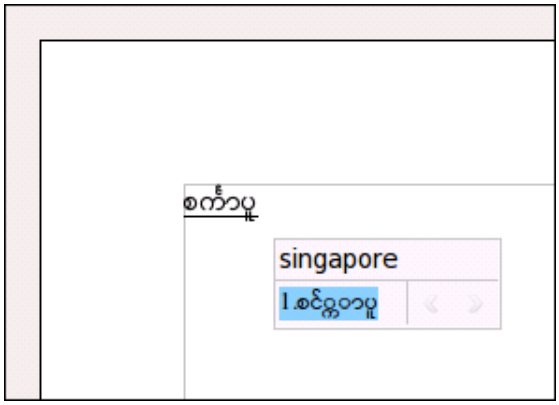
This section details a number of issues that are known to occur with WaitZar. If you find any new issues, please submit a bug report online:

http://code.google.com/p/scim-waitzar/issues/entry/

5.1. Pat-Sint Choices are Glitched

Even if you follow the advice in Section 2.4, you will find that some words are still incorrect when presented as choices. For example, below is a screenshot of “singapore” as it appears in the lookup table –note that the actual, typed version is still correct.

Figure 10. The Myanmar text is “decomposed” in SCIM, but renders correctly in Open Office.



The WaitZar team is uncertain as to what is causing this glitch. There are many interacting components (cairo, pango, GTK, or even scim itself) that might be slightly off. Perhaps the issue is actually a fontconfig issue on our end. We *could* solve this problem by defaulting SCIM to the Zawgyi-One font, but we feel that this solution would be a cheap hack that delays the adoption of Unicode 5.1 for Burmese in Linux. Instead, we will write some test cases to determine what package is causing the glitch, and then work with those developers to resolve this problem.

5.2. Alt+Tab Sometimes “Sticks”

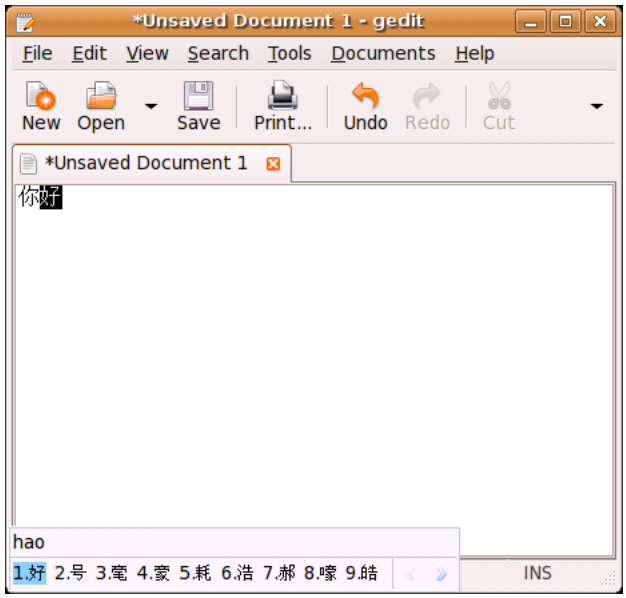
This appears to be a bug in SCIM itself; under some configurations, switching to WaitZar will cause the “Alt +Tab” combination to “stick” –pressing this will outline the window one is hoping to switch to, but releasing it will see you at the original window; nothing happens. Clicking on any other window will cause this bug to go away until the next time you switch languages.

So far, we have only found one setup where this occurs, and this behavior is apparent in other IMEngines (Chinese, Thai, etc.), so it is probably a bizarre case of SCIM and Linux interacting poorly. Please report this bug if you encounter it, and we will try our best to narrow it down and fix it.

5.3. Choices Stuck in the Lower-Left

Under some configurations (usually, the same ones described in Section 5.2), the list of choices will appear “locked” to the lower-left of the current window. See the screenshot below for an example of this annoying bug, which makes typing very difficult. Of course, you can click and drag the choices window to a more convenient location, but it will jump right back to the lower-left corner as soon as you switch the language again.

Figure 11. The choices window is "sticking" to the corner of gedit. This occurs with all language IMEs (Chinese is shown here).

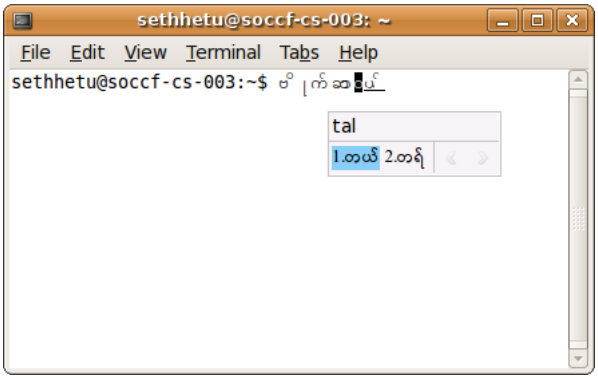


We might try re-positioning the choices window to somewhere nicer when WaitZar gets a "focus_in()" message, but seeing as how this is tied in to the previous bug, we decided not to hack around it until we were sure it couldn't be fixed normally.

5.4. The Console Looks Glitchy

Have a look at the following screenshot.

Figure 12. Notice how the text does not compose correctly in the console.



If this is what you see when you try to type WaitZar in the console, there's no need a file a bug report; we're aware. If you actually see proper Burmese, then *please* let us know how to configure the console to make this work. (**Note:** using Zawgyi-One or Win Innwa is *not* the solution we are looking for.)

A. License

scim-waitzar is licensed under the GNU General Public License v3, although some parts of it are licensed under the Apache License v2, and a small amount is licensed under the GNU General Public License v2.

This means that **scim-waitzar** is Open Source, of course, and free software as well. The original WaitZar on Windows is licensed under a BSD-style license; we chose to license **scim-waitzar** under the GPL to make it more compatible with the GNU philosophy, and chose version 3.0 to help strictly enforce that this software is not used in a non-open way.

Of course, our concerns apply only to Linux, really. The WaitZar library API, available from:

http://code.google.com/p/waitzar/downloads/list

...contains the majority of the encapsulated functionality; both the Windows *and* the Linux project build primarily off of this library, adding only implementation-specific code. Furthermore, the library is released under the Apache License, with the intention of not forcing any restrictions on developers.

Discussion about WaitZar is generally taken with both appreciation and seriousness, and occurs primarily on the mmgeeks forum, in English:

http://mmgeeks.org/forum/

B. How It Works

This section explains the use of trigram models in language modeling. It is currently under development.