# SToNe - Song-Tool by Nelphindal

Nelphindal

July 24, 2015

created by LaTeX

# Contents

# 1 Introduction

This manual is like the toolbox itself under construction. Feel free to notify me if you find a bug or find a already written passage is unclear. On page 22 you can find some solutions for popular problems.

# 2 Prerequisites

To execute this toolbox you have any version of the Java runtime environment, short form JRE, of at least version 7.x [1] installed. After custom installation you can execute this tool by double clicking on it (Launcher.jar).
The toolbox itself can be downloaded from my git-repository currently at `https://github.com/Greonyral/stone/SToNe.jar`. Currently supported operating system are Windows XP, Vista, 7 and 8. All distributions based on current versions of Unix should be able to run the toolbox.

[1] Download for example at `http://www.java.com`

# 3 Overview

There are currently mainly 4 tools included into this toolbox:

1. AbcCreator: GUI for Bruzo's BruTE
2. SongbookUpdater: Fast update of the Songbook.pluginData for the Plugin by Chiran
3. FileEditor: A collection of tools to edit abc-files
4. VersionControl: A basic git client, used to distribute songs and keep them up-to-date

On every start the toolbox will ask you, which of these tools shall be executed. Every checked tool will be checked if available and up-to-date, and if not the newest version will be downloaded automatically. To start the main-tool you can either double-click on the Launcher.jar or invoke it manually by calling java.exe[2].

[2]On windows you have to type the full path to the Java runtime if it is not included in your PATH-variable. A valid command would be
'C:\ProgramFiles\java\jre7\java.exe -jar Launcher.jar'

# 4 Features

What this toolbox can do:

1. Check for new version of this tool, download it and replace the currently used file by the new one
2. Create the local git-repository
3. Download changes from local git-repository
4. Upload committed changes to local git-repository
5. Write for any known accounts the needed file for Songbook-Plugin by Chiran
6. Provide a GUI for the use of BruTE
7. Provide a mechanism for encryption of abc-files before upload them and decryption on downloading
8. A simple client to read and write messages
9. Reset the repository and delete created files by this tool. The repository band however will be kept.
10. Repair - Delete files influencing the behaviour of this tool, including the local repository - only on confirmation
11. Uniform songs - Apply a name scheme to every selected song
12. Modify song titles
13. Use SSH instead of HTTPS - normally only of interest on Unix

In the following each chapter will explain every feature. The features described at VersionControl are included currently only in the builds for our band A Rock and A Hard Place.

## 4.1 The config file

The basic settings can be changed in the config-file for the tool. It can be found under Windows at

%USERPROFILE%\SToNe\launch.cfg.txt

under Unix it is

~/.SToNe/launch.cfg

To edit open this file with any text editor. These are the section with the options you can edit:

[*global*]

name
Your name, will be used to set the name in created abc-files and as name for the commits to create.

path
By default the path to the directory created by LoTRO where your screenshots, settings and music are.

[*vc*]

branch
The branch to work on. This key will not appear unless you add it there. It is currently named

launch.cfg.txt

and can

board
Value of the field to run the Bulletin Board. Will be used to enable it on next run if set to true.

aes-key
The key to use for encryption and decryption of the protected songs.

login
The user-name for your github.com account.

pwd
The password for your github.com account.

ssh_url
The url to use when using the ssh-protocol instead of default https to transfer data between local and remote repository.

use_ssh
The switch to turn on permanently ssh-protocol.

[*brute*]

player
The path to an executable of the AbcPlayer. When set every time you hit Test, the player will start to play the created song.

style
The style being set on last run. And will be used on next run.

[*fileEditor*]

scheme
The entry to use for the option to uniform the song-titles. See page 20 for more details.

# 5 Getting new versions

There are basically two ways to get new versions. First one is to download it again with the same links like on page 4. Second way is to let the toolbox do this itself. The toolbox will contact the repository and download the new version of selected tools automatically. The tool will check the installed Java version and notify you if it detected that there is a new version available[3].

[3]Works currently only with installed Java by Oracle

# 6 VersionControl

Due to various song-files we had, i wrote this tool originally to make it easier to maintain our songs. The original idea was to have a simple tool, hiding all unnecessary options of the used git-protocol. Because of this, all actions of this tool are reflecting options a git-client would offer, too. But however this tool has only a subset of the options of a full git-client. This allows you to use any git-client if you feel that this tool is not supporting the operation you want to do. But before this you should read at least one short introduction of git to get an idea what you are doing.

## 6.1 Basics of git - A short glossary

I will use some terminology of git. Therefore this will be a short introduction. Git is a version control. Normally it is used for programmers to distribute code, apply patches, build releases and roll back changes if necessary. Since source-code and the abc-files are more or less the same, i.e. text-files, it is a nice way to publish our songs, making it very simple to keep everyone up-to-date. So if you use git your actions will be (normally) always revocable. But enough of this let us start to get the basics done. So here are the basic concepts.

### 6.1.1 Repository

A copy of a hierarchy within a directory with its files. To transfer data between two different repositories

```
git fetch
```

to get data from a repository to the local one and to transfer from it to remote one

```
git push
```

are used. After you fetched the head of a remote repository the files in your local repository has not been changed already. Only the identifaction of the remote head has been stored into a variable called

```
FETCH_HEAD
```

this will be used with one of the following commands. To get the changes either a merge, checkout-command or reset-command is needed. The difference between the commands is that

```
git merge
```

will have the effect that two trees or branches will be seen as one after completion,

```
git checkout FETCH_HEAD
```

will switch between your current branch and the newly fetched tree or branch and will warn you overwriting files where you might lose changes whereas

```
git reset FETCH_HEAD --hard
```

will enforce it and after completion the state of your local repository will be identically to the one of the remote repository. The combination of fetch, checkout and merge is the combination which will be used most often and therefore a command exists doing it all at once:

```
git pull
```

it does a fetch, checking it out and merging with current branch.
You can identify the root directory of a git-repository by the existence of a directory called

```
.git
```

Deleting this directory will delete all data linked to this repository. Edit the files and directories contained in this directory only when you have a very good idea of what you are doing. It might cause the repository to be no longer usable afterwards if you mess it up.

### 6.1.2 Commit

A commit is a snapshot of repository, also known as revision. Each commit has a short hash, commits with same hash are considered equal. Same files with same content can result in different hashes and therefore commits.

### 6.1.3 Tree

Series of commits are contained of one or more branches. The older commit is the parent, the younger one the child. It can be happen that there are multiple parents or children. In this case there are multiple trees of one branch. Git has no problems with multiple children. But in our case, the existence of multiple children will cause sooner or later the merging of both trees to one commit with multiple parents. This will be done every time contacting the remote repository. In case the program can not merge changes automatically you will get an alert and you will not be able to upload any commits until you fixed the merge-conflict. Conflicts can be avoided as long everyone is modifying only the own directory. After a successful merge both branches are equal, all following commits are contained in every previously merged tree.

### 6.1.4 Branch

But you can define also two different trees which you will never want to merge, or at least not for the moment. In this case every not merged tree has to be checked in a branch. Technically it is not more then a named pointer to a top of a tree. By default there is always a branch, called master. This tool uses two branches. The branch

    master

is containing the band directory.

### 6.1.5 Head

The head is the newest commit of a branch and its tree.

### 6.1.6 State

Git knows several states of file depending on the difference to the latest commit of current branch. Some are acquired passively, other states can be reached only by invoking certain git commands. The first states are the passive ones.

**New** is every file not been in the repository before

**Changed** is every file with different content compared to last commit

**Missing** is every missing file compared to last commit

The states of commits are as follows.

**Added** is every new file to be committed

**Modified** is every changed file to be committed

**Deleted** is every file to be removed on next commit, or on foreign commit the file which has been removed.

## 6.2 Install git

There are various implementations of git. Search for

> git windows

will find some results. There is not the best solution but a nice overview over good GUIs for git is `http://git-scm.com/downloads/guis`. Get the one looking good for your needs, the included installers should set everything up to work. If you can not decide what you need, get GitExtensions[4], it will cover everything you will ever need.

## 6.3 Create local repository

By default the local repository is Music/band within the base path you selected. The tool will create the band directory if it does not exist and the option to

> Download songs

is set, initialize the repository and start the download. It can take several minutes depending on your downstream-speed.
You can do it manually by invoking

```
git clone https://github.com/Greonyral/lotro-songs.git
```

This needs a git installation, you can find a guide to set it up on page 11

[4] `http://sourceforge.net/projects/gitextensions/`

## 6.4 Download to local repository

This option is executed each time you select the VersionControl. The tool will get the head from the remote repository and will compare it with the one in the local repository. Any changes between the remote and local commit will be displayed if the option to show the changes is active. In any case the remote commit is not equal the local branch will be set to be equal to the remote branch. Local not yet uploaded commits will be based on the new head. An error message will be displayed if the attempt failed. The tool will ask for files missing in the current head if they shall be restored. Encrypted files will be decrypted automatically. It might be that you have to enter a key, if is first time you use this option. For more information about decryption see the chapter on page 13.

## 6.5 Upload from local repository

Uploading changes consists of two parts. First one is to create a snapshot, this is making a commit in terms of git. Second part is to upload this snapshot to a remote repository. Both parts will be done by enabling the option to

> Commit

for you. All you need is a log-in on the platform providing the remote repository[5]. The registration is for free. After this the owner of the repository (this is me) has to add your account to have write access. For this a simple mail with your github login is enough. The upload will be done by the tool-box after every successful commit. You can enable the option to commit, select no file and let the upload to be done in case the commit is successful but the upload fails.

## 6.6 Commit changes

This option requires the user-name and password entered. You will be asked only for those files found in

`band/<your name>`

The tool will show you each song created, changed or missing and will ask you if you want to add it. To prevent future name clashes the files with identical file names will be highlighted. You can not use the GUI to commit these files. You can abort this progress at any point, the added files will be remembered and included next time you finish a commit. Currently the GUI does not support the undo option of adding the files.
To remove committed file(s) you can see page 23 for details.

## 6.7 Reset

This will undo several actions of the local repository. All tracked files will be rolled back to last commit. Any files added, or deleted, for next commit will be unstaged. This option should unblock any failing download.

[5] www.github.com/greonyral/lotro-songs

## 6.8 Encryption and Decryption

Files can be encrypted before uploading them and the tool will decrypt such files automatically on downloading[6]. You can preselect encryption by naming the file to end with

```
.enc.abc
```

or select it during the commit process. The encrypted files are placed at

```
band/enc/<your name>
```

as indication for the tool that these files needs to be decrypted on downloading. The decrypted files will be put with normal **.abc** ending into the normal directory. On first time using this feature you will need to enter the key.

## 6.9 SSH

SSH is a protocol for cryptographically secured internet connections (instead of HTTPS). It allows login at github without password and user-name. Since the support is not that well on Windows this options should be used only on Unix systems. The default URL for SSH for the currently used repository is

```
git@github.com:Greonyral/lotro-songs.git
```

At page 6 it is described how to change it (for example to use a different ssh-key).
Using a complex ssh-config will require a full installation of ssh and git.

[6]The used alogrithm is AES with a 256 bit key

# 7 AbcCreator (a GUI-Wrapper of BruTE)

Of the included tools this one is the largest. It includes a complete runtime of Bruzo's BruTE and offers a - hopefully - easy to use GUI to adjust the settings.

## 7.1 Style

This option passes its value to the underlying BruTE-runtime. It influences the naming of the parts. For now valid styles are

`Rocks`

the default style,

`Meisterbarden`

a style used by Meisterbarden of Bree and

`TSO`

the style used for Starlight-Orchestra.

## 7.2 Settings on start-up

Most of the settings offered by this tool are optional.

### 7.2.1 Test with Abc-Player

You can provide the path to a runtime of Abc-Player to test your created abc. For this select

`AbcPlayer.jar`

or however you renamed it. The abc will start to play automatically hitting

`Test`

as soon it has been created.

### 7.2.2 Specify additional drum-maps

To import custom drum-maps all maps have to be placed in the same directory and fit into the name scheme. This is

```
DRUM ID SUFFIX
DRUM ::= drum
ID ::= [0-9][0-9]*
SUFFIX ::= .drummap.txt
```

or put it in other words: 'drum' any positive number followed by '.drummap.txt'.

### 7.2.3 Specify custom midi to abc mapping

You can change the default mapping of the midi-instruments to their counterparts in the abc. The map representing the default mapping can be found in the repository next to the Launcher.jar.

## 7.3 Settings in the loop

After the initialization is done, the tool will enter a loop asking for the midi-file to transcribe. The created abc-file will created as the midi-file-ending replaced by abc. Hitting abort will abort the tool and terminate it, even if other tools have been selected for execution.

### 7.3.1 Visualization of the midi

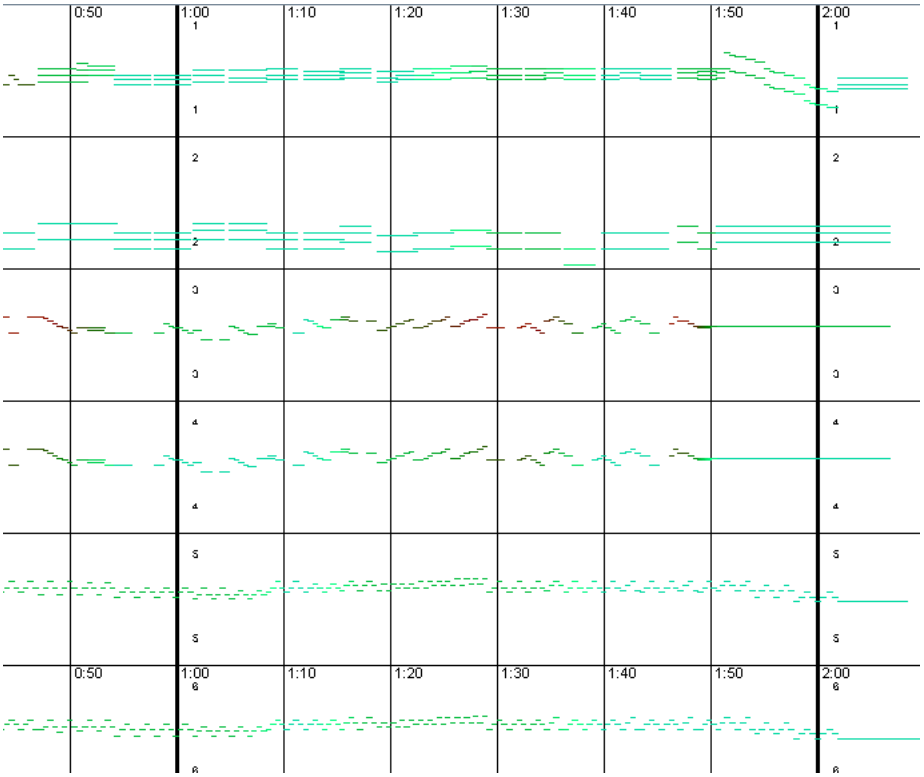


Figure 1: Tracks of a midi-file displayed

As soon the tool read the selected midi-file a second window will appear, visualizing the parsed midi-file.
The colors are representing the volume as encoded in the midi-file. It goes from blue for very quiet over green and orange to red for very loud. The lines are showing that there is a note active played at this time.

### 7.3.2 Setting for this abc

The main window will display every non empty track to left side, every instrument of LoTRO to right, i will call them categories. The bottom-most category is special, it is indicating that the tracks linked with it shall not be played, or that the instrument of the midi-file was not mapped to a category by default. The instruments in center represent the later band setting. On pointing with your mouse on tracks every instrument playing this track will be marked yellow, instruments and tracks playing the same instrument type will be marked orange.

The color scheme is same while pointing on the instruments and categories. You can load previously saved maps.

### 7.3.3 Set instrument to play a midi track

It will be the mainly setting you will change. You can create new instruments by dragging the midi-track to the category you want a instrument from. Empty instruments will be removed. You can track multiple tracks to same instrument if you want to have a instrument to play multiple tracks.
To have a midi-track played by more than one instrument of same category, the button

`Split`

has to be pushed down. No association to an instrument will be deleted as long the bottom is pushed down and all instruments belong to same category.
In case you want one midi-track played by multiple instruments of different categories, you have to create a clone of the midi-track for each different category. Right click on the track to clone will create a new track to link to another category than the original track. Linking the clone to the empty category will remove it again.


Figure 2: Selected cloned Track 1 and played by Lute. Track 3 played by Lute, too. Original Track 1 played by Harp. Track 2 is assigned for being played by Theorbo and therefore not being highlighted.

### 7.3.4 Song global settings

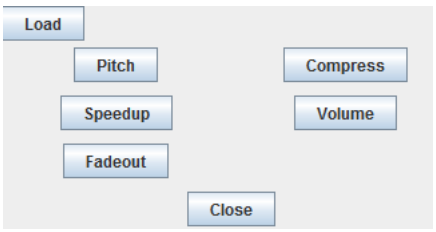


Figure 3: Available Settings valid for entire song

You can edit the global settings - volume, speed-up - for this songs by clicking on the settings button on the bottom of the displayed window. The title for song will be set after clicking OK.

### 7.3.5 Track local settings


Figure 4: Showing Settings for Track 1

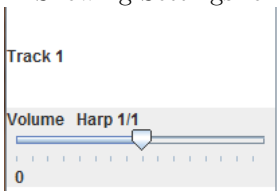


Figure 5: ... and after clicking on Volume

By left clicking a track the options for this track will appear, this are volume, pitch and delay. Select the option to change by clicking on the field and adjust the value with the sliders.

### 7.3.6 Instrument local settings

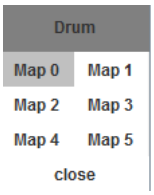


Figure 6: Available maps for the drum. Default map 0 is selected.

Currently there is only one setting you can change for each instrument. This is the map for drums. Clicking on it will show you a dialogue to change the map. The default map 0 will be used by default as long no drum-map is highlighted.

### 7.3.7 Example

The tool created with the default mapping a lute for a part. In the end you want to have two flutes playing it. On of them is already there, the other one is missing. The following two steps may be swapped. First you drag the track to the already existing flute. The now empty lute will be removed from the center and on hovering with the mouse over the flute or track will show you that the

track is played now by the flute. On testing you notice that this track is to fast for only one flute and it would be better to have only every second note be played by this flute and the other half of notes to be played on the second flute. So you enable the split mode and drag the track to the flute of the category side. A new flute will appear in the center and on hovering over the track will mark now both flutes.

### 7.3.8 Load a saved map

I tried to make the loading of the maps as simple as possible. Most valid maps should work. There are no explicit checks of the map you are going to load, you will notice on the result if something went wrong. To start the loading click on the

`Load`

bottom on the bottom of the main window and select in the appearing window the map to load. The tool will remove all links and depending on the speed of your machine you will see a moving progress bar in the top. Wait until the bar disappears and a message is displayed saying that the load is completed or failed.

**Please note:** There is no mechanism to recover from errors encountered during loading a map. Because of this the GUI will be locked and only thing you can do is to close it.

# 8 FileEditor

This tool offers some options to edit one or more songs. After enabling this option the tool will analyse all selected songs and in most cases it can extract the different components of the title like information about the instrument, duration, last date of modification. These different components may be reordered, restructured by using the concept of the name scheme described later in this section. This allows for example to add the information about duration to songs where this information is missing. Or to display the instrument for every song between any kind of brackets, braces or similar.

## 8.1 Reset modDate

This option can change the date of last modification of every file being checked in into a git-repository. By default the git-repository is expected to be

`band`

located within

`Music`

## 8.2 Uniform Titles

The option to uniform the song titles runs after selecting the files completely automatically. It tries to identify all needed parts of the specified name-scheme and applies it afterwards. However some files might be not identified correctly. To minimize the risk of data-loss all corrected files will be copied maintaining the internal hierachy to

`Music_rewritten`

See the chapter about Name-scheme on page 20 for specifying the new style of the title lines.

## 8.3 Modify titles and numbering

In case you want to change the numbering or titles of songs this option will help you. It tries to analyse the title of the selected song(s) and shows you a dialogue after it to change it. The new title line will be written as specified in the name scheme you had to enter on start-up.

## 8.4 Side effects of uniforming and modifying

A side effect of the options using the name-scheme is that the parts of the songs will be ordered by their index number (the number at the line starting with X:), the number you have to type to play this part.

## 8.5 Name scheme

It determines the style of the new created title lines. There are variables and constants. This to may depend on the line number. *Variables* may be different for each songs and each of its lines. **Constants** are constantly the name for each song. And some parts can appear only for some lines. By default the name scheme is as follows
*title index/total* [*instrument*] (*duration* ) *date*
The *date* will appear only in the second line and the *duration* in the first line of each song. The *date* itself is composed of *month* and *day*. The required entry for the name scheme to get this as result would be

```
%title %index/%total [%instrument]$1{ (%duration)}$2{ %b %d}
```

You can edit your own name scheme. The *variables* are

%title
Its the part the tool could identify as title equal over all parts. In case there is no title the name of the abc will be used.

%index
Its any number the tool could find in the title identifying this track. In case there was no number to identify the track number will be used. In this case it will be the number you have to use in game to select this track.

%total
Number of identified parts. A part does not count in case its title has been identified with more than one index.

%duration
The duration of the midi. Will be calculated automatically if the tool can not find any in the title.

%m
The month (1-12) of last modification.

%b
The month (Jan-Dec) of last modification.

%d
The day (1-31) of last modification.

%y
The year (00-99) of last modification.

%instrument
The instrument(s) for this part.

%Instrument
The instrument(s) for this part, starting with capital letter.

%INSTRUMENT
The instrument(s) for this part, with capital letters only.

The $-symbol with following number or a list of numbers seperated by ',' defines the line dependent components. These have to put between '{' and '}'. Any other symbol will be interpreted as a **constant**.

# 9 Repair

Deletes the config-file and the local repository. On next run the config-file will be created with default values. To prevent data-loss the deletion of the directory hosting the local repository has to be confirmed. The directory created for the bulletin board will be deleted.

# 10 Troubleshooting

## 10.1 You found a bug

Send me a report, please. Try to describe it as good as you can to make it easy for me to find it. I will try to fix it, if you did it well enough. Mail it to greonyral@arcor.de or to nelphindal@gmail.com.

## 10.2 The tool is always asking to add one or more files, which you do not want to add

Open the directory containing the local repository. You are in correct directoy if you can see a directory called

```
.git
```

If you can find a file called

```
 .gitignore
```

open it with any text editor and append a line with the file(s) you want to ignore in future. The paths have to be relative to the directory you are in.
In case the file does not exist it is getting a bit harder since Windows normally do not know files with no file ending and a leading dot. Create a new file, write the file(s) with their relative path(s) into it, one file per line and safe it to make it easier for you on the Desktop for example 'ignore.txt'. Now open the command line, navigate to the Desktop with

```
cd Desktop
```

and enter

```
move ignore.txt .gitignore
```

This will rename the 'ignore.txt' to '.gitignore'. Only thing to do now is to cut-and-paste it into the directory.

## 10.3 You deleted an file you want to restore

Git might help you if you checked this file in before you deleted it and committed it: the tool will detected the missing file when you run VersionControl next time and ask to restore it. While commit is active make sure to answer first question, to remove the file from the repositories, with no.

## 10.4 You get an error on merging

It should never happen, but if it happened to you try to reset the local repository into a valid state. For more information see the paragraph about how to reset**??**.

## 10.5 You want to create a shadow copy of your repository

You can tell the tool that the repository somewhere else by changing the base of the path. See the chapter for the config-file6 on how to edit the file for your needs.

## 10.6 You uploaded a file you do not share under any circumstances

In this case you have to wipe your commit. Its basically descriped in the next paragraph. To prevent uploading again you can put this file in to your ignore-list. For more information on how to do this see page22

## 10.7 You uploaded a subset of changes by accident

No panic. Even deleted songs are normally not lost. A benefit of git is to save all previous commited states of your repository. You can try to fix it yourself on your own risk or ask me for help to recover it. This operation is a

```
git reset
```

command. There are easy how-tos in the web.

## 10.8 Purge a commit

This operation is rewriting the history and should be never done. But there might be reasons to do so. If you do it alone, inform EVERYONE what happened, that we have a change to wipe our history if we had the contaminated commit.