SSTBuilder Users Manual

Overview

SSTBuilder is a cross platform GUI based tool used to assist users in configuring and building the SST software.

SSTBuilder does not eliminate the pre-requisite requirements necessary to build SST (such as Boost, and MPI), instead it provides a more user friendly display of all the available configuration options for SST. Additionally, specific configuration options can be saved as a profile and easily restored allowing the user to easily handle multiple configurations.

Installation and Running

SSTBuilder is a separate application that is included as part of the SST distribution, and is located in a sub-directory below the root of the main SST directory.

SSTBuilder requires a MacOSX or Linux based machine that is running the native graphical windows manager.

The source code for SSTBuilder is located at <sstroot>/tools/sstbuilder

Pre-compiled versions of SSTBuilder are located in the <sstroot>/tools/sstbuilder/release_linux and <sstroot>/tools/sstbuilder/release_macosx. The precompiled versions were built using Ubuntu 12.04 and Mac OSX 10.8.4. In both release directories, any support files are provided as necessary.

Linux Distribution:
- The linux distribution is compressed into a zip file named sstbuilder_linux.zip
- A starting profile is provided named “startup.sbdr”. This provides a preliminary starting profile.
- Cross platform shared object files (Qt libraries) are provided. The user must set their LD_LIBRARY_PATH to point to <sstroot>/tools/sstbuilder/release_linux/lib. This is usually done in the .bashrc file. Also the qt.conf file is configured to load any Qt plugins located in the <sstroot>/tools/sstbuilder/release_linux/plugins directory.
- The binary file sstbuilder was built using 64 bit Ubuntu Linux v12.04, gcc v4.6.3 and Qt v5.0.2

Mac OSX Distribution:
- The Mac distribution is compressed into a zip file named sstbuilder_macosx.zip
- A starting profile is provided named “startup.sbdr”. This provides a preliminary starting profile.
- The framework sstbuilder.app is available and built for Mac OSX 10.8.4 The sstbuilder.app is a framework (directory) that contains everything necessary (program binary, icons, shared libraries, etc) to run the application.
- The sstbuilder.app was build using 64 bit OSX 10.8.4, clang v425.0.28 and Qt v5.0.2

If the pre-compiled versions do not run properly, the user should build the application for their system. Instructions are included below.

Linux Compile:
- Download and install the Qt 5.0.2 (or newer) Development Kit located at http://qt-project.org/downloads and perform a standard install.
- Run the Qt Creator development interface, and load the sstbuilder.pro file. Then compile the SSTBuilder application.
- To find out the shared object dependencies for the sstbuilder, cd to the directory containing the sstbuilder, and run > ldd sstbuilder (using a terminal).

Mac OSX Compile:
- Download and install the Qt 5.0.2 (or newer) Development Kit located at http://qt-project.org/downloads and perform a standard install.
- Qt for Mac requires Apple xcode development suite to be installed from http://developer.apple.com/xcode
- Run the Qt Creator development interface, and load the sstbuilder.pro file. Then compile the SSTBuilder application.

- To find out the shared object dependencies for the sstbuilder.app, cd to the directory containing the sstbuilder.app, and run
  >otool –L sstbuilder.app/Contents/MacOS/sstbuilder (using a terminal).
- To finalize the sstbuilder.app framework, cd to the directory containing the sstbuilder.app, and run
  >macdeployqt sstbuilder.app (using a terminal). This will merge the required shared objects into the sstbuilder.app framework so that the framework can be delivered as a standalone application. Note: macdeployqt is a Qt provided application and must be in the users PATH environment for proper operation.

General Usage Concepts

- All dependencies (external packages) for building SST must be properly compiled and installed on the system. These pre-requisites include: GNU Libtools, Boost, MPI (OpenMPI or MPICH), DRAMSIM2, Gem5, QSim, etc. See https://code.google.com/p/sst-simulator/wiki/SSTBuildAndInstall3dot0series for more information.
- SSTBuilder provides a convenient and graphical method for performing the SST configuration, compile and install, but does not eliminate the requirements for setting appropriate environment path variables.
- SSTBuilder allows for various SST setup profiles to be quickly selected by storing environment variables and SST configuration options as a profile. Profiles can be modified and saved to files and restored as necessary.
- When SSTBuilder starts up, it will load the default profile (.sstbuilder file in the users home directory).
- Any changes to the SSTBuilder setup will be automatically written to the default profile. Additionally, the user can save the current setup as a separate profile. This saved profile can then later be loaded to restore the setup.
- Any profile that is loaded will then become the default profile.
- Environment groups are a collection of environment variables that can be enabled/disabled as a group. Environment groups are created/edited in the setup dialog. Environment groups allow for quick configuration changes to different versions of dependent libraries. For example one Environment group can be configured for OpenMPI whereas a separate group can be setup for MPICH.
- When SSTBuilder runs autogen.sh, configure or make, it will load the enabled Environment groups for that particular run.

Initial Startup and Usage

1) When SSTBuilder is started for the first time, the program will have no loaded profile, and will look as shown (Figure 1)

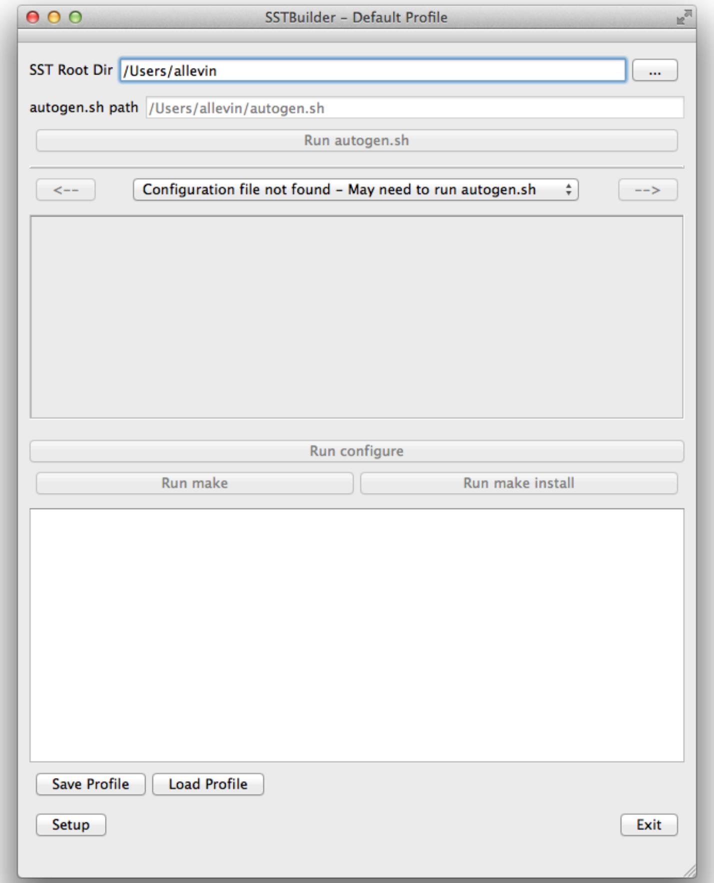
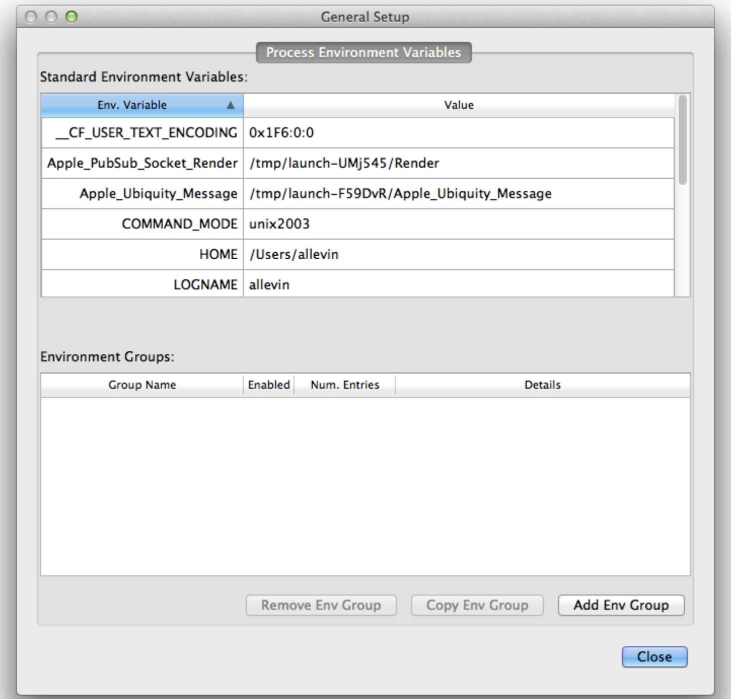


Figure 1

2) If the user clicks the "Setup" button, the dialog will show the setup dialog with unpopulated Environment Groups (Figure 2).



Standard Environment Variables:

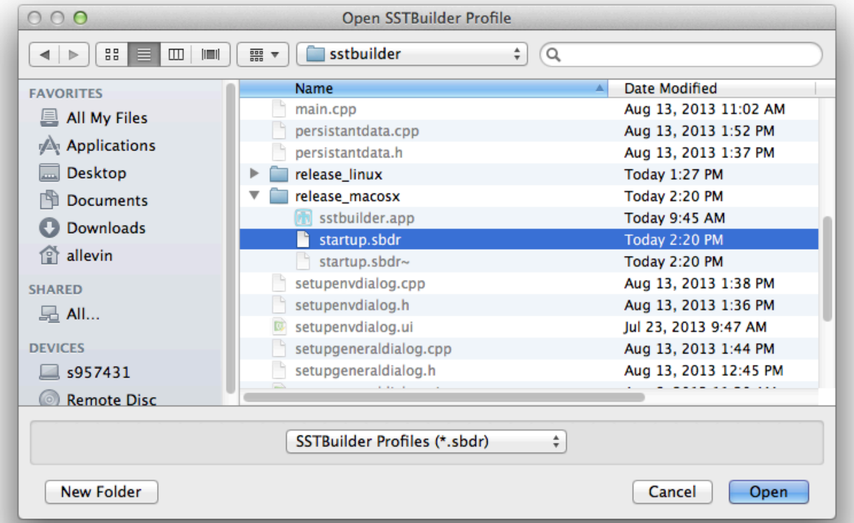
Env. Variable	Value
__CF_USER_TEXT_ENCODING	0x1F6:0:0
Apple_PubSub_Socket_Render	/tmp/launch-UMj545/Render
Apple_Ubiquity_Message	/tmp/launch-F59DvR/Apple_Ubiquity_Message
COMMAND_MODE	unix2003
HOME	/Users/allevin
LOGNAME	allevin

Environment Groups:

Group Name	Enabled	Num. Entries	Details

Figure 2

3) The user should load the starting profile by clicking the “Load Profile” Button and finding the “startup.sbdr” file in the located in the same directory as the application (Figure 3).

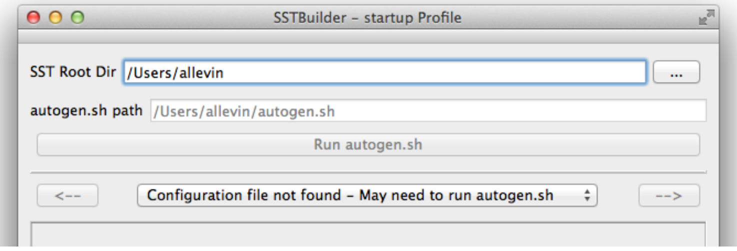


Name	Date Modified
main.cpp	Aug 13, 2013 11:02 AM
persistantdata.cpp	Aug 13, 2013 1:52 PM
persistantdata.h	Aug 13, 2013 1:37 PM
release_linux	Today 1:27 PM
release_macosx	Today 2:20 PM
sstbuilder.app	Today 9:45 AM
startup.sbdr	Today 2:20 PM
startup.sbdr~	Today 2:20 PM
setupenvdialog.cpp	Aug 13, 2013 1:38 PM
setupenvdialog.h	Aug 13, 2013 1:36 PM
setupenvdialog.ui	Jul 23, 2013 9:47 AM
setupgeneraldialog.cpp	Aug 13, 2013 1:44 PM
setupgeneraldialog.h	Aug 13, 2013 12:45 PM

SSTBuilder Profiles (*.sbdr)

Figure 3

4) After loading the profile, the title of the application will change to show what profile is currently active (Figure 4).


SSTBuilder – startup Profile

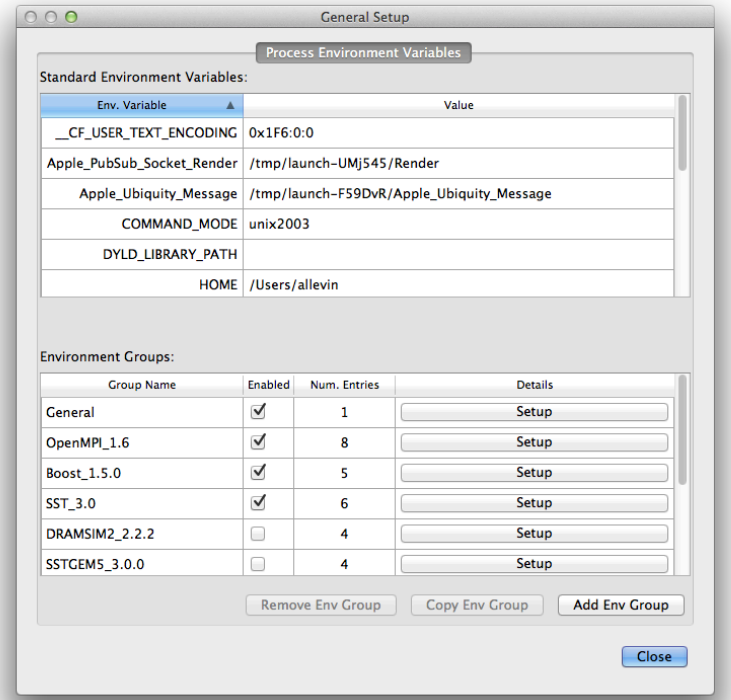
SST Root Dir: /Users/allevin

autogen.sh path: /Users/allevin/autogen.sh

Configuration file not found – May need to run autogen.sh

Figure 4

5) If the user clicks the "Setup" button, the dialog will now show the setup dialog with populated Environment Groups for that configuration (Figure 5)


General Setup

Process Environment Variables

Standard Environment Variables:

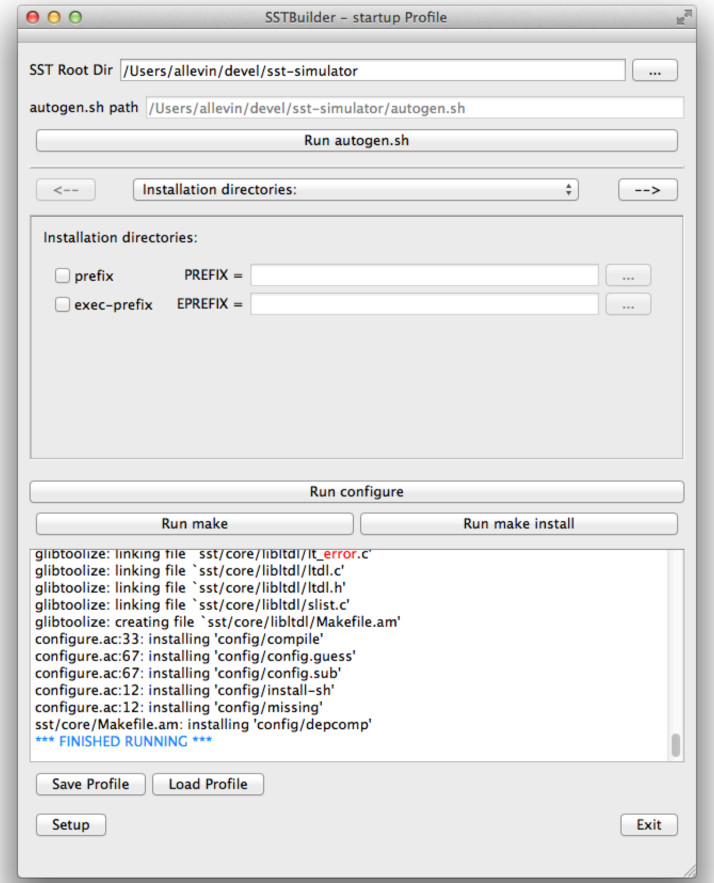
Env. Variable	Value
__CF_USER_TEXT_ENCODING	0x1F6:0:0
Apple_PubSub_Socket_Render	/tmp/launch-UMj545/Render
Apple_Ubiquity_Message	/tmp/launch-F59DvR/Apple_Ubiquity_Message
COMMAND_MODE	unix2003
DYLD_LIBRARY_PATH	
HOME	/Users/allevin

Environment Groups:

Group Name	Enabled	Num. Entries	Details
General	✓	1	Setup
OpenMPI_1.6	✓	8	Setup
Boost_1.5.0	✓	5	Setup
SST_3.0	✓	6	Setup
DRAMSIM2_2.2.2		4	Setup
SSTGEM5_3.0.0		4	Setup

Figure 5

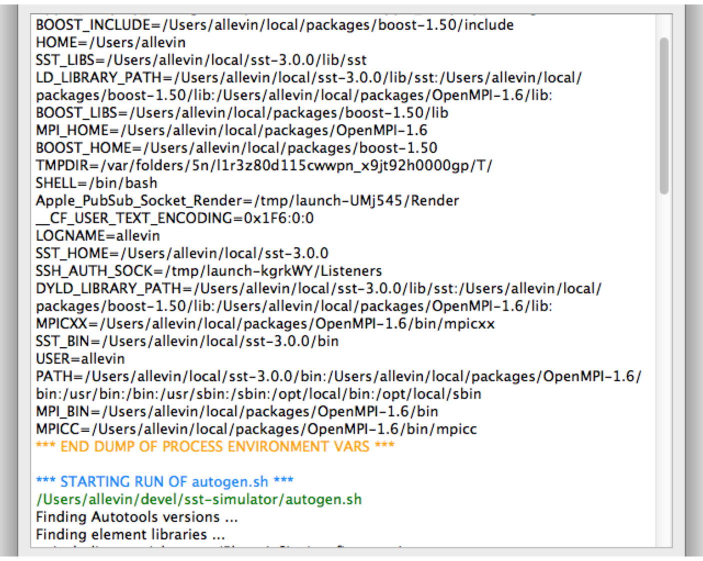
6) The user must select the SST Root directory by either entering the path or clicking the "…" button. This directory must be where the autogen.sh file is located. When configured properly, the "Run autogen.sh" button will be enabled and the autogen can be run to generate an SST configuration file as shown (Figure 6).


SSTBuilder – startup Profile
SST Root Dir
/Users/allevin/devel/sst-simulator
...
autogen.sh path
/Users/allevin/devel/sst-simulator/autogen.sh
Run autogen.sh
<--
Installation directories:
-->
Installation directories:
prefix
PREFIX =
exec-prefix
EPREFIX =
Run configure
Run make
Run make install
Save Profile
Load Profile
Setup
Exit

```
glibtoolize: linking file `sst/core/libltdl/lt_error.c'
glibtoolize: linking file `sst/core/libltdl/ltdl.c'
glibtoolize: linking file `sst/core/libltdl/ltdl.h'
glibtoolize: linking file `sst/core/libltdl/slist.c'
glibtoolize: creating file `sst/core/libltdl/Makefile.am'
configure.ac:33: installing 'config/compile'
configure.ac:67: installing 'config/config.guess'
configure.ac:67: installing 'config/config.sub'
configure.ac:12: installing 'config/install-sh'
configure.ac:12: installing 'config/missing'
sst/core/Makefile.am: installing 'config/depcomp'
*** FINISHED RUNNING ***
```

Figure 6

7) Note that after the output window displays a dump of the Environment variables used for the process, and the process (in this case autogen.sh) output (Figure 7).

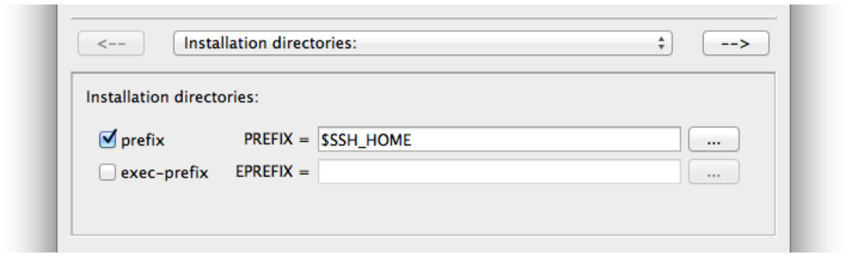


```
BOOST_INCLUDE=/Users/allevin/local/packages/boost-1.50/include
HOME=/Users/allevin
SST_LIBS=/Users/allevin/local/sst-3.0.0/lib/sst
LD_LIBRARY_PATH=/Users/allevin/local/sst-3.0.0/lib/sst:/Users/allevin/local/
packages/boost-1.50/lib:/Users/allevin/local/packages/OpenMPI-1.6/lib:
BOOST_LIBS=/Users/allevin/local/packages/boost-1.50/lib
MPI_HOME=/Users/allevin/local/packages/OpenMPI-1.6
BOOST_HOME=/Users/allevin/local/packages/boost-1.50
TMPDIR=/var/folders/5n/l1r3z80d115cwwpn_x9jt92h0000gp/T/
SHELL=/bin/bash
Apple_PubSub_Socket_Render=/tmp/launch-UMj545/Render
__CF_USER_TEXT_ENCODING=0x1F6:0:0
LOGNAME=allevin
SST_HOME=/Users/allevin/local/sst-3.0.0
SSH_AUTH_SOCK=/tmp/launch-kgrkWY/Listeners
DYLD_LIBRARY_PATH=/Users/allevin/local/sst-3.0.0/lib/sst:/Users/allevin/local/
packages/boost-1.50/lib:/Users/allevin/local/packages/OpenMPI-1.6/lib:
MPICXX=/Users/allevin/local/packages/OpenMPI-1.6/bin/mpicxx
SST_BIN=/Users/allevin/local/sst-3.0.0/bin
USER=allevin
PATH=/Users/allevin/local/sst-3.0.0/bin:/Users/allevin/local/packages/OpenMPI-1.6/
bin:/usr/bin:/bin:/usr/sbin:/sbin:/opt/local/bin:/opt/local/sbin
MPI_BIN=/Users/allevin/local/packages/OpenMPI-1.6/bin
MPICC=/Users/allevin/local/packages/OpenMPI-1.6/bin/mpicc
*** END DUMP OF PROCESS ENVIRONMENT VARS ***

*** STARTING RUN OF autogen.sh ***
/Users/allevin/devel/sst-simulator/autogen.sh
Finding Autotools versions ...
Finding element libraries ...
```

Figure 7

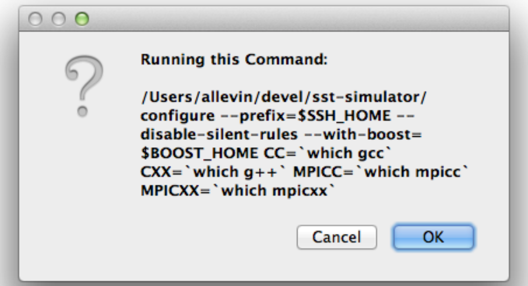
8) Once the autogen.sh has been run, all of the various configuration options can be set as desired (Figure 8).



```
[✓] prefix        PREFIX =  $SSH_HOME
[ ] exec-prefix   EPREFIX =
```

Figure 8

9) Once all the configuration options are selected, the "Run configure" button is pressed. This will generate a dialog previewing the configure command settings (the user can use this to verify their configuration) (Figure 9).

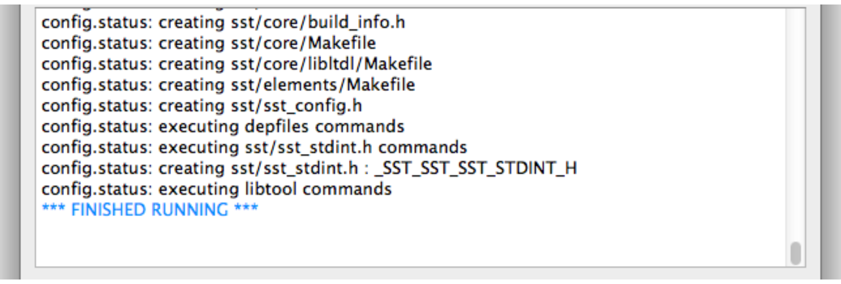


Running this Command:

```
/Users/allevin/devel/sst-simulator/
configure --prefix=$SSH_HOME --
disable-silent-rules --with-boost=
$BOOST_HOME CC=`which gcc`
CXX=`which g++` MPICC=`which mpicc`
MPICXX=`which mpicxx`
```

Figure 9

10) After accepting the preview dialog, SST will be configured. As before, the environment variables and output from configure will be displayed in the output window (Figure 10).

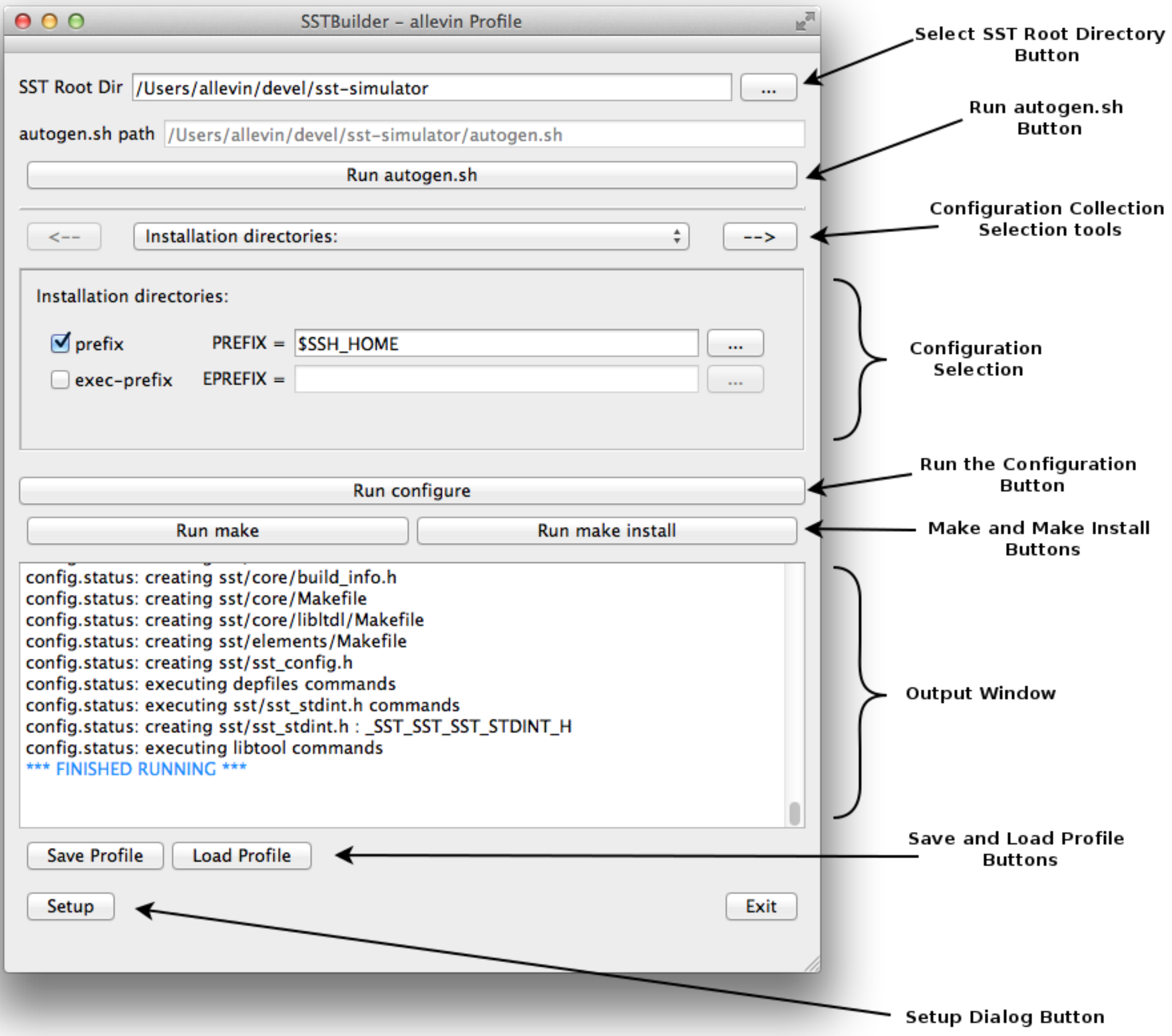


```
config.status: creating sst/core/build_info.h
config.status: creating sst/core/Makefile
config.status: creating sst/core/libltdl/Makefile
config.status: creating sst/elements/Makefile
config.status: creating sst/sst_config.h
config.status: executing depfiles commands
config.status: executing sst/sst_stdint.h commands
config.status: creating sst/sst_stdint.h : _SST_SST_SST_STDINT_H
config.status: executing libtool commands
*** FINISHED RUNNING ***
```

Figure 10

11) Finally, if configure was successful and a Makefile was created, the user can click either the "Run make" or "Run make install" as desired.

Main Screen

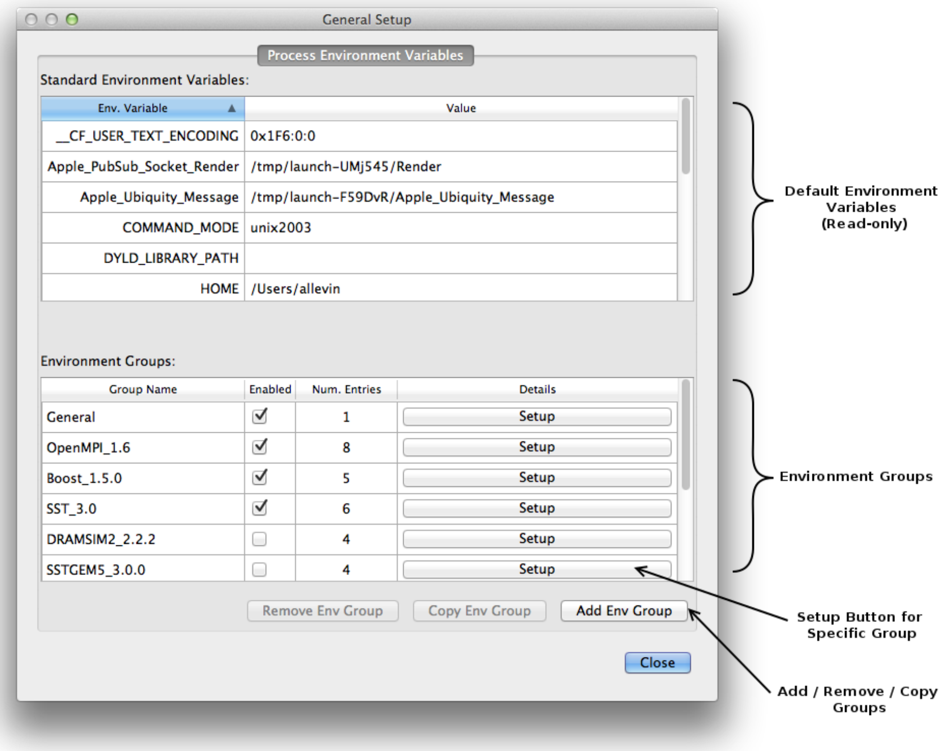

SSTBuilder – allevin Profile
SST Root Dir
/Users/allevin/devel/sst-simulator
...
autogen.sh path
/Users/allevin/devel/sst-simulator/autogen.sh
Run autogen.sh
<--
Installation directories:
-->
Installation directories:
prefix
PREFIX =
$SSH_HOME
exec-prefix
EPREFIX =
Run configure
Run make
Run make install
Save Profile
Load Profile
Setup
Exit
Select SST Root Directory Button
Run autogen.sh Button
Configuration Collection Selection tools
Configuration Selection
Run the Configuration Button
Make and Make Install Buttons
Output Window
Save and Load Profile Buttons
Setup Dialog Button

```
config.status: creating sst/core/build_info.h
config.status: creating sst/core/Makefile
config.status: creating sst/core/libltdl/Makefile
config.status: creating sst/elements/Makefile
config.status: creating sst/sst_config.h
config.status: executing depfiles commands
config.status: executing sst/sst_stdint.h commands
config.status: creating sst/sst_stdint.h : _SST_SST_SST_STDINT_H
config.status: executing libtool commands
*** FINISHED RUNNING ***
```

- In the main window, the configuration area is dynamically populated from the configuration file created by autogen.sh. There will be a number of groupings that can be switched by the arrow buttons and/or the pull-down control.
- The current selections are:
    - Installation directories: – Where to install SST.
    - Fine tuning of the installation directories: – Finer grain control over the SST installation directories.
    - Program names: - Modifications to the program names
    - System types: - Settings for various system types
    - Optional Features: - Settings for optional SST features.
    - Optional Packages: - Settings for optional SST Packages.
    - Some influential environment variables: - Various complier and other settings.
- Clicking on a configuration setting checkbox will enable/disable that option.
    - Most of the configuration settings have a entry field for additional information (such as targeted home directory for SST). Some of these entry fields must be populated. These required fields will be marked with “<Required>” when the configuration setting is enabled.
    - Entry fields that are meant for directories, will contain a button “…” which will allow the user to select a specific directory if desired. The user can also manually enter a path or use an environment variable (example $SST_HOME).

        - Entry fields can also contain environment variables or execute strings (example `which gcc`) to provide entries in the fields that will be passed to the system shell when the run button is pressed.
- The configuration settings and entries are saved as part of a profile. When creating a new configuration file (by running autogen.sh), the program will attempt to do its best to match up old configuration choices with the updated configuration file.
- The user can right click on the output window to get the following options:
    - Select All – Select all text
    - Copy – Copy the text to the clipboard
    - Clear – Clear the output window
    - Print – Print the contents of the output window

<u>Setup Dialog</u>


General Setup
Process Environment Variables
Remove Env Group
Copy Env Group
Add Env Group
Close
Default Environment Variables (Read-only)
Environment Groups
Setup Button for Specific Group
Add / Remove / Copy Groups

Standard Environment Variables:

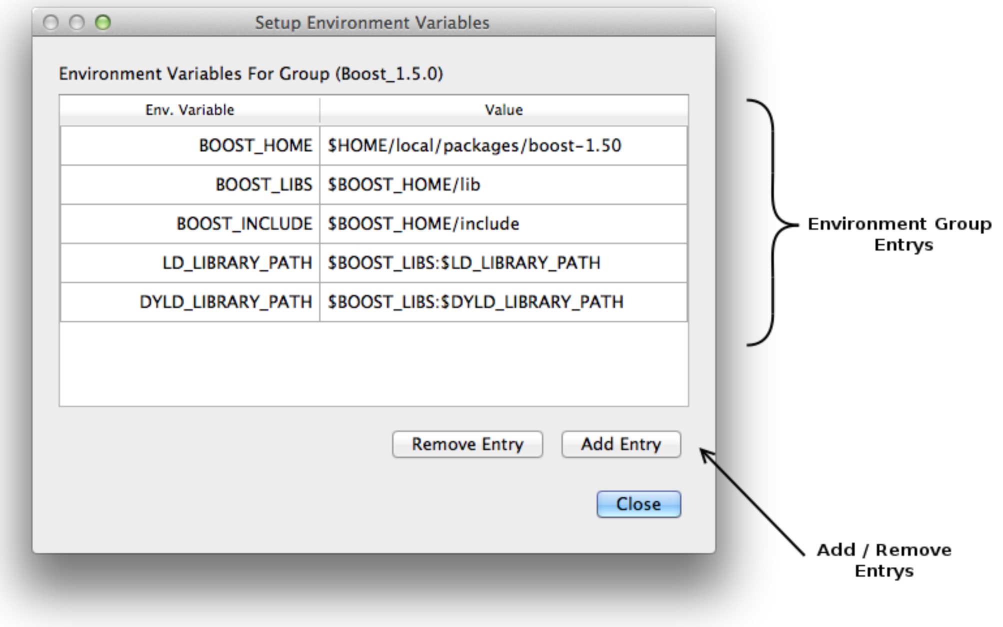
Env. Variable ▲	Value
__CF_USER_TEXT_ENCODING	0x1F6:0:0
Apple_PubSub_Socket_Render	/tmp/launch-UMj545/Render
Apple_Ubiquity_Message	/tmp/launch-F59DvR/Apple_Ubiquity_Message
COMMAND_MODE	unix2003
DYLD_LIBRARY_PATH	
HOME	/Users/allevin

Environment Groups:

Group Name	Enabled	Num. Entries	Details
General	✓	1	Setup
OpenMPI_1.6	✓	8	Setup
Boost_1.5.0	✓	5	Setup
SST_3.0	✓	6	Setup
DRAMSIM2_2.2.2	☐	4	Setup
SSTGEM5_3.0.0	☐	4	Setup

- The default environment variables are variables provided by default by the system.
- And environment group is a collection of environment variables (assigned to a Group Name) that can be collectively enabled/disabled.
- The purpose of environment groups is to allow multiple environment settings to live side by side. This allows the user to quickly select what options they want. For example, one group could be for OpenMPI, whereas another group could be for MPICH. Another example is for one group to be Boost Version 1.5.0 and another for Boost Version 1.5.4. In both of these examples, the user can quickly enable one or the other groups to select the operation of the SST build.
- Environment groups have a Group Name and can be individually enabled. To add a group, press the “Add Env Group” button. The new group will be added to the bottom of the list and a unique group name should be entered.
- To edit the group’s environment variables, press the setup button on the right of the Group Name entry.
- Environment groups can be copied (for quick duplication of a similar group) or removed with the “Copy Env Group” and “Remove Env Group” buttons.

<u>Environment Group Setup Dialog</u>


Setup Environment Variables

Environment Variables For Group (Boost_1.5.0)

Env. Variable	Value
BOOST_HOME	$HOME/local/packages/boost-1.50
BOOST_LIBS	$BOOST_HOME/lib
BOOST_INCLUDE	$BOOST_HOME/include
LD_LIBRARY_PATH	$BOOST_LIBS:$LD_LIBRARY_PATH
DYLD_LIBRARY_PATH	$BOOST_LIBS:$DYLD_LIBRARY_PATH

- The environment group entries allows the user to add/remove entries assigned to a specific Environment group.
- To add an entry, press the “Add Entry” button. The new entry will be added to the bottom of the list. The name and value for the Environment variable can then be entered.
- Note that and environment variables (such as $PATH or $LD_LIBRARY_PATH) can be used the value field.

<u>Logging</u>

- When the “Run autogen.sh”, “Run configure”, “Run make” or “Run make install” buttons are pressed. An appropriate time stamped log file will be created in the selected SST root directory.