

Developing Triana: Source Code from Subversion, Setup and Build

Matthew Shields*

Abstract

This document describes the steps in correctly checking out the source code for the core Triana and Triana toolboxes from the SVN repository and building from the source code.

If you don't know what Triana is, don't know how to use a SVN client¹, then it is recommended that you download the latest packaged version of Triana from the website <http://www.trianacode.org/projects/triana>.

1 Conventions

Note: These instructions are aimed at command line SVN users. Windows or similar GUI based users should be able to follow the instructions by using the repository and module names within their particular SVN Client.

- Text written *this font* should be typed as command line input. Commands should be entered without line breaks unless explicitly instructed.
- *Text written in this font* and surrounded by < ... > should be replaced by the users appropriate details.
- **Text in this font** signifies the unix command line prompt, the text following it will be the command to type.

2 Repository Root and Passwords

All repositories have a common root <https://svn.trianacode.org/triana>, just add the package name to get to the files. Viewing this root in a normal Web browser will enable you to browser the source code online. Usually you will either want the main trunk or the stable branch. Anonymous read only access is provided to the core Triana code and the default toolboxes. If you require “commit” rights you should contact the main Triana developers via the mailing list or direct via the contact points on the website.

3 A Word About Directory Structure

For administration reasons Triana is split into a number of different packages which are stored in various repositories. Some of these are project specific and not public so don't assume that because something is listed in this document that you will be able to get the source code from the SVN server.

*matthew.shields@astro.cf.ac.uk

¹See <https://svn.trianacode.org> for a list of SVN clients

The public packages are:

1. **trianacore** – The Triana Environment itself.
2. **trianatools** – The default toolboxes that come with Triana.
3. **toolboxes-dev** – The unstable toolboxes that developers are currently working with. Use at you own risk.

The project specific packages are:

1. GEO. The Gravitational Waves tools.
2. Gravity. Example tools from the book “Gravity From The Ground Up” by Bernard Schutz.
3. GriPhyN. Tools from collaboration work with the GriPhyN project.

There are two standard ways to structure a Triana distribution from source, dependant on: whether you expect to be making regular changes to the source code under your control and/or you want to do frequent updates for the latest version from SVN; Or you just want to check out the latest version and build it once. SVN allows checking modules out inside other modules which will give the same structure as the packaged version of Triana but it will complain if an `update` or `commit` is attempted in the source tree where there is a foreign module lower down the tree.

4 Easy Install

These instructions will checkout and install Triana to the same structure as the packaged release version. It is fine for most users however if you expect to use SVN for more than updating small numbers of files it is suggested you use the other set of instructions.

This install will put all of the various toolboxes inside the Triana source tree and they will need to be removed to perform a `svn update` on the core Triana code and then checked back out again.

From the command line:

```
$ svn co https://svn.trianacode.org/triana/trianacore/trunk/ triana
$ cd triana
$ svn co https://svn.trianacode.org/triana/trianatools/trunk/ toolboxes
```

Note: The rest of the Triana modules from SVN are optional.

```
$ cd toolboxes
$ svn co https://svn.trianacode.org/triana/toolboxes-dev/trunk/ toolboxes-dev
```

After those SVN commands you should have a directory structure like this:

```
triana/
  /bin/
  /toolboxes/
```

```
/Audio
/... other standard toolboxes
/toolboxes-dev
```

The toolbox structure is the import thing, there will be extra directories under `triana/` not mentioned here.

4.1 Build and Run

To build set an environment variable for your system, `$TRIANA` for Unix like systems or `%TRIANA%` for Windows, to point the top level `triana` directory. Run the build script, `buildTriana` for Unix or `buildTriana.bat` for Windows:

```
$ $TRIANA/bin/buildTriana
```

Run the start script, `triana` or `triana.bat`:

```
$ $TRIANA/bin/triana
```

5 Developer Install

If you are intending to write or modify any code within the various SVN modules or you just want to regularly update the modules from SVN. Then we suggest that you keep all the SVN modules in their own separate directory structures. The *Ant*² build file is written to be able to build from default either the previous structure or a directory structure where all the SVN modules are at the same level in the file system.

For instance a typical developer directory structure looks like this:

```
project/triana/
project/toolboxes/
project/toolboxes-dev/
project/toolboxes_other
project/toolboxes_other/GEO
project/toolboxes_other/GravityFromTheGroundUp
project/toolboxes_other/GriPhyN
```

So from a sensible starting directory run the SVN commands from section 4 without the `cd` commands so that the Triana core and default toolboxes modules from SVN reside in your current directory.

Note: The GEO, Gravity and Griphyn toolboxes should really reside in a separate subdirectory. Here we have created a directory called “ `toolboxes_other`”, this is because Triana assumes that a toolbox contains packages, so GEO for instance is the top level package for the GEO tools. When Triana attempts to locate tools it uses the following path :-

```
[toolbox][tool package]*[toolname]
```

e.g. `[/project/toolboxes/] [SignalProc/] [Input/] [Wave]`

²<http://ant.apache.org/>

or [/project/toolboxes_other/] [GEO/] [Algorithms/] [SaturationMon]

The build and run instructions are almost the same as for the easy install.

- Set the TRIANA environment variable to point to the triana directory.
- Either edit the build file (build.xml in the main Triana home directory) and set the appropriate toolbox location variables in the “User Editable” section, or the preferred method add a new file called “build.properties” to the Triana home directory with the properties and their values, you can edit and rename the example file “example.build.properties”. The contents of a typical “build.properties” file is listed below

```
# STANDARD JAVAC FLAGS
-----
javac.flag.failonerror=true
javac.flag.deprecation=off
javac.flag.debug=on
javac.flag.optimize=on
javac.flag.verbose=off

# JVM Heap Sizes
-----
jvm.heap.initial=-Xms128M
jvm.heap.max=-Xmx256M

# JVM Versions
-----
jvm.source=1.3
jvm.target=1.3

# GAP, P2PS & WSPEER LOCATIONS
-----
p2ps.base.dir=../p2ps
gap.base.dir=../p2ptoolkit
wspeer.base.dir=../wspeer

# TOOLBOX LOCATIONS
-----
# triana.geo.tools=../toolboxes-other/GEO
 triana.gravity.tools=../toolboxes-other/gravity
# triana.griphyn.tools=../toolboxes-other/GriPhyn
 triana.release.dir=/Users/spxmss/physics/release/
 triana.release.version=triana-3.2
 triana.release.tag=Triana_3_2_Release
```

- Run the buildTriana script.

Note: Unlike the standard installation, Triana will not automatically pick up the toolboxes when it is started so you will have to add those within Triana from the menu *Tools, Edit Tool Box Paths*. In this example set up, select the directories - `/project/toolboxes/`, `/project/toolboxes-dev/`, `/project/toolboxes_other/`.

6 Other Install

You can actually have your toolbox modules anywhere you like in your file system and still have the build pick them up and compile them. In the file `build.xml` there is a commented section titled “USER EDITABLE PROPERTIES” within this section there are a number of commented out properties for the various toolbox modules. Uncomment any of these and set them to the appropriate location to override the default locations. Alternatively add them to the “`build.properties`” file with the appropriate lines containing – `propertyname=value`.