

## NIGHTLY BUILD SYSTEM CONSOLIDATION

```
.../src/svn/openlaszlo/trunk/docs/developers/developers.preface.ht
[ava] Aha! Using role='lzx-embednew' on file program-development-$1.lzx and paramete
[ava] Got canvas-width of 500 and canvas-height of 40
[ava] Aha! Using role='lzx-embednew' on file program-development-$8.lzx and paramete
[ava] Got canvas-width of 500 and canvas-height of 100
[ava] Aha! Using role='lzx-embednew' on file program-development-$9.lzx and paramete
[ava] Got canvas-width of 500 and canvas-height of 200
[ava] Aha! Using role='lzx-embednew' on file program-development-$10.lzx and paramete
[ava] Got canvas-width of 500 and canvas-height of 200
[ava] Aha! Using role='lzx-embednew' on file program-development-$11.lzx
[ava] Got canvas-width of 500 and canvas-height of 100
[ava] Aha! Using role='lzx-embednew' on file program-development-$12.lzx
[ava] Got canvas-width of 500 and canvas-height of 100
```

BEN SHINE

DECEMBER 19, 2007



# Consolidate or Suffer!

## Current problems

Some of the known problems in the current nightly build system:

- A) Susceptible to hardware failures: one of the three of build machines fails every couple of months, causing about a day's work to diagnose and rebuild the machine. This unplanned downtime and fixing can be very disruptive.
- B) Building a native installer for windows and the mac requires both a working windows build machine or a working mac build machine.
- C) Coarse control: we can only trigger a complete build; we don't have hooks to redo just part of a build. In particular, fetching the source from svn is a long process that has to be redone if anything went wrong in the build at all.
- D) Poor visibility of results: figuring out whether a build succeeded or failed, and what status it's in, requires examining logs on all three build machines, and checking for the presence of files on download.openlaszlo.org.
- E) Poor visibility of status: during a build, to find out how it's doing, three machines and download.openlaszlo.org must be checked.
- F) Our sense of "success" and "failure" combines pure-code issues (did the unit tests fail? did the server compile?) with network issues (did the ssh connection stay up for long enough to sftp the entire 30M distro?) with hardware issues (did linux-builder's disk fill up during the build?)
- G) Execution time: doing a complete build takes at least an hour and forty minutes; only one build can be done at once.
- H) Difficulty with repeating previous builds: we do not currently have a versioning system for environment requirements. When things change that require different versions of jars in \$ANT\_HOME/lib, or different versions of python or java, that is not succinctly captured by our process. Old builds are therefore difficult to repeat.
- I) Performance testing is not done as part of the nightly build system.
- J) Browser-based testing is not done as part of the nightly build system.
- K) Most knowledge about building the NSIS (Windows) installer has evaporated from the team. Also, it uses a five-year-old version of NSIS (<http://www.nullsoft.com/free/nsis/version-history.html>)
- L) Very strange sh and bat files (lzc.bat) to invoke the compiler as a command line tool.

## A Partial Solution

*One ring to rule them all!*

The OpenLaszlo platform is entirely based on Java, with the exception of a few little-used command line tools (lzc/lzc.bat) which have already been customized for each host OS. The only thing we use native tools for, in the nightly builds, is to build the *installers*. We would only have to use one architecture to build things if only we could generate native installers for a variety of OS's from one builder.

Several software tools do just that:

- Macrovision InstallAnywhere [<http://www.macrovision.com/products/installation/installanywhere.htm>],
- install4j [<http://www.ej-technologies.com/products/install4j/features.html>],
- BitRock [<http://bitrock.com/>].

I propose that we build the platform on just one build machine, and use one of these tools to generate installers for each OS we want to target. This will free us from having to have **three machines up and running** in order to generate an official release; it will **speed up build time**, by **obviating the need to duplicate source** and build products on several machines; it will **simplify build status monitoring** by only needing to check one log.

Disadvantages:

- This **won't tell us if we break the build** only on a particular OS. I think that's okay: maintaining the build-ability of OpenLaszlo on several development platforms is a lower priority than reliably producing distributions that run on several platforms.
- There will be a significant **cost**, in dollars and time, to buying and configuring a multi-platform installer-builder. My hope is that planning for and executing this improvement will quiet the need for build-machine-related fire-drills.
- It might not work. I'm particularly concerned that we might not be able to get the single installer-builder to generate **correct relative windows paths** for documentation. If we reach a dead end with one of these, another one will probably work, or we can figure out workarounds. Again, my hope is that the cost even of false starts will be addressed by the decreased maintenance and monitoring costs.

**This is the way we sweep the porch**

1. cron job kicks off a build on linux-builder
2. linux-builder sucks down the source from svn
3. linux-builder does a standard "build" target
4. **Milestone: the build compiled successfully**
5. linux-builder runs the unit tests (ant megatest)
6. **Milestone: the build built and the unit tests passed**
7. linux-builder builds the docs
8. **Milestone: the doc built successfully**
9. Use our current tools to make the non-native distros: servlet-only, source, linux, devkit. Begin pushing these to [download.openlaszlo.org](http://download.openlaszlo.org).
10. Deploy the webapp on [labs.openlaszlo.org](http://labs.openlaszlo.org)
11. **Milestone: build deployed on labs.**
12. Invoke the multi-installer builder to build installers for the mac and windows.
13. Push the native installers to [download.openlaszlo.org](http://download.openlaszlo.org)
14. **Milestone: build complete.**

## Next Steps: Separation of Concerns

**When resources are available to work on the build system, we should endeavor to create a federation of processes which are each responsible for a separate concern within our nightly maintenance milieux.**

The build system is trying to address each of these concerns:

- A) Are each of the machines in the build workflow alive and reachable?
- B) Does the current source compile?
- C) Does the current source pass the command-line unit tests?
- D) Does the current build pass the browser-based unit tests in each browser x each runtime x each platform?
- E) Did the nightly distributions build successfully for each target platform?
- F) Is the current build successfully deployed on labs.openlaszlo.org?
- G) Is tomcat currently running on labs.openlaszlo.org?

### **Getting there from here**

We can move towards this separation of concerns by teasing apart convoluted processes one or two at a time. Right now, the pain point is maintaining three build machines, so that is where we should focus our energy. Uniformity of implementation is much to be desired, but creating a uniform solution would require tossing out most of what we have, centralizing on one technology, which feels like a heavier change than we can currently afford.

### **Very Next Step:**

The next most-crucial-thing, after simplifying the nightly builds to just require one machine to build native installers, is to automate running the smoketest and other unit tests in each browser, in each platform, in each runtime, for each build.