Kepler - Bug #4311

Build system appears to lock if there is a conflict upon update

08/13/2009 11:48 AM - Chad Berkley

Status:	Resolved	**Start date:**	08/13/2009
Priority:	Normal	**Due date:**	
Assignee:	David Welker	**% Done:**	0%
Category:	build system	**Estimated time:**	0.00 hour
Target version:	2.X.Y	**Spent time:**	0.00 hour
Bugzilla-Id:	4311		

Description

When you do an 'ant update' if a conflict is encountered by svn, the build system appears to lock up. Probably need to detect this state and tell the user.

History

#1 - 08/26/2009 02:36 PM - Matt Jones

Conflicts are common when multiple people work on a file. This needs to get fixed for 2.0. Need to provide a way for people to provide input to answer how to resolve the conflict.

#2 - 10/29/2009 01:18 PM - Chad Berkley

pushing to post 2.0 since this does not affect the user experience.

#3 - 01/07/2010 07:24 PM - David Welker

The solution to this problem is not in the build system. You should do the update manually if you have made changes.

#4 - 01/07/2010 07:57 PM - Oliver Soong

I'm not so sure it's a good idea to ignore this. This happened to Mark, Regetz, and me. None of us had made any local changes, although I can't remember exactly why there was a collision. I think it was some config file that had a default stored locally in the repository and changed when Kepler started. In any case, it was in no way clear whether, how, or why the build was stalled. A simple solution would be to pass stdin, stdout, and stderr to the console.

I understand there's some complication to using stdin. Maybe this will help some:
http://www.coderanch.com/t/419646/Ant-Maven-Other-Build-Tools/java-program-accept-user-input

#5 - 01/07/2010 08:12 PM - David Welker

Are you saying that when the update failed to complete, it did not say why? That is unusual. Can you reproduce this error and share the output that you get?

Normally, when an error occurs updating (for example, due to a lock) it is quite clear what caused the problem. For example, as referenced in bug 4281, when the "ant update" does not complete due to a lock, you get an error that explains why:

```
===========
[update-modules] Updating ptolemy-lib...
[update-modules] svn -r head update c:\tmp\cxh\src\kepler\ptolemy-lib
[update-modules] At revision 20190.
[update-modules]
[update-modules] svn -r head update c:\tmp\cxh\src\kepler\build-area
[update-modules] svn: Working copy 'c:\tmp\cxh\src\kepler\build-area' locked
[update-modules] svn: run 'svn cleanup' to remove locks (type 'svn help
cleanup' for details)
[update-modules]
[update-modules] WARNING: It appears that the command did not execute properly
and exited with an exit code of: 1 ===========
```

If you can reproduce this issue and verify that in fact the conflict does not show up in output, I will look into it more. Otherwise, the solution is to fix the problem manually and run "ant update" again.

#6 - 01/08/2010 09:52 AM - Oliver Soong

The problem is that it never fails to complete. Normally on a conflict, svn outputs some information and blocks for user input. For whatever reason (wrong stream or whatever), ant doesn't pass this information along and the whole thing just sits there. There's no error message, no prompt, and no activity.

If you want to reproduce this from head, pick a file, update to the previous revision, figure out a line that changed, change it to induce a conflict, then run ant update. The whole thing blocks. I used RExpression.java, and this is the output:

```
[update-modules] Updating r...
[update-modules] svn -r head update C:\Documents and Settings\soong\Desktop\kepler\r
```

This lasts until I terminate (Ctrl-C).

#7 - 01/08/2010 10:15 AM - Chad Berkley

This bug has happened to me exactly the way that Oliver has described on many occasions. It should be fixed prior to 2.0.

#8 - 01/08/2010 12:38 PM - David Welker

I was curious what the basis of your opinion that this should be done before 2.0 is? This is functionality to make life a little more convenient for developers, it does not affect the user experience. Previously, you had written: "pushing to post 2.0 since this does not affect the user experience."

I am happy looking into this now, but I was just curious about the change in opinion.

#9 - 01/08/2010 12:45 PM - Chad Berkley

I still don't think it really affects the user experience, but it's happened to me enough times to be annoying so I changed my mind on the timeframe. It seems easy enough to fix. If it's too hard, we can still push it to post 2.0.

#10 - 01/08/2010 01:41 PM - David Welker

A couple of things. First, there are two alternative things we could mean by "fixing" this issue: (1) Print out the error so that the user can resolve the issue manually. (2) Allow the user to interact with svn to resolve the conflict while the update command is temporarily suspended.

Of the two solutions, (2) would be preferable in some situations (if the conflict is easy to resolve). But it would be rather challenging, because you would have to find a way to "come back" after the conflict is resolved. Keep in mind that one way people resolve conflicts is by jumping into emacs or vi or whatever. I am not enough of a UNIX maven to know whether the output you see on these screens is part of the standard out stream or not. Is it possible to suspend the update while the user has full rein to do whatever they like from the command-line? Maybe it is, if there is a standard message (or messages) that SVN gives when a conflict is resolved, I suppose one could monitor standard out and resume only when that message is received. What if the user decides to resolve the conflict in another command-line window? Then your program will be suspended forever, and you wouldn't realize it -- until some random time when the message that you are monitoring standard out for somehow appears, and the "missing" update command unexpected resumes, doing who knows what.

Further, is this really a desirable work process for many conflicts? Users may need to investigate how to resolve the conflict. It could be a complicated task unto itself and users might be switching back and forth between multiple programs, making phone calls and writing emails to resolve the conflict. It is possible that the update could be suspended for quite a while during this phase, and the later modules are updated in a way that is incompatible with the previous modules. In fact, this is a risk with any update now: it just isn't likely, because the "ant update" command usually executes fairly quickly.

On the whole, I think there are many potential issues with trying to allow the user to suspend the update while they try to figure out how to fix a conflict. What if it takes a long time and therefore the modules are updated in inconsistent states as a result? What if they resolve the conflict in a different command-line window, thereby depriving the update command knowledge of when the conflict is resolved?

Do we really want to be mixing updates and resolving conflicts in one huge process? Maybe, but it sounds like it may be more trouble than it is worth.

Approach (1) in contrast is fairly simple. The update fails. You fix the conflict. You may or may not then choose to finish the update. When you do finish the update, all of the modules that were updated before the conflict are updated again, ensuring consistency. You are completely free to fix the conflict in an entirely different command-line window than you do the update and you are also able to use multiple command-line windows safely.

Yes, it is annoying when the command fails because you have to fix something manually. But it is also annoying when the command is suspended and you have to fix something manually. At least if the command fails, you have less issues with consistency and the possibility of indefinite suspension -- and remember, the indefinite suspension might be ended much later when you resolve a totally unrelated conflict -- meaning that updates that you think are very undesirable at that moment could occur without you expecting it.

Now, if I understand correctly, even if we go with solution (1) it has been suggested that solution is not currently working, since there is some sort of locking occurring and there is no output whatsoever indicating the source of the problem. The update command just mysteriously stops working? Is my understanding correct?

#11 - 01/08/2010 01:49 PM - Chad Berkley

If an error is printed, I'd be fine with it. Right now, it appears to hang and does not give any further output.

#12 - 01/08/2010 02:16 PM - Oliver Soong

I'm not clear why option 2 is so complicated. Just pass stdin, stdout, and stderr from svn to ant. I don't expect anything more.

#13 - 01/12/2010 08:24 PM - David Welker

Fixed. Now, modules that have local modifications are not updated and an appropriate WARNING is given.

As far as Oliver Soong not seeing why option 2 is complicated, he should feel free to modify the build himself according to his views. I don't think option 2 is a good idea, but I could be wrong.

As far as I am concerned, this bug is fixed. However, instead of closing it, I am reclassifying it as an enhancement (since the build no longer freezes when conflicts occur) and reassigning it to Oliver, who can then either close the bug or try modifying the build according to his ideal view.

#14 - 01/12/2010 10:15 PM - David Welker

I have looked into the CommandLine.java class more with Oliver Soong, and am going to try to hook process so that one can pass input to them when appropriate.

Right now, CommandLine.java is only designed to handle output. But, it may make sense for it to handle input as well...

#15 - 02/11/2010 04:31 AM - David Welker

This was fixed by Chad in an even better way by taking advantage of command-line options that exist in the svn update command. For some reason, we forgot to close this bug.

#16 - 03/27/2013 02:26 PM - Redmine Admin

Original Bugzilla ID was 4311