Kepler - Bug #5065

In shared or administrative installations, the ability to store modules.txt and extra modules locally. This way, the module manager will work smoothly on Windows without having to run as an administrator.

06/30/2010 08:42 PM - David Welker

Status:	Resolved	**Start date:**	06/30/2010
Priority:	Normal	**Due date:**	
Assignee:	David Welker	**% Done:**	0%
Category:	general	**Estimated time:**	0.00 hour
Target version:	2.2.0	**Spent time:**	0.00 hour
Bugzilla-Id:	5065		

Description

(1) In shared or administrative installations, the ability to store modules.txt and extra modules locally. This way, the module manager will work smoothly on Windows without having to run as an administrator.

Problem: The file modules.txt is read by Kepler to determine which modules are active. Both modules.txt and local modules are currently stored in the installation area, which in shared installations is read-only. Because extra modules are executed, security might dictate that in shared installation situations modules.txt not be changed and new modules not be downloaded. This was the original rationale for going forward with our current design However, on Windows, the default for installations appears to be to install as an administrator. As a result, even in private non-shared installations on Windows, it can be difficult to use the Module Manager. To do so, a user must remember to run as an administrator, which gives Kepler the ability to write to the installation area.

Solution 1: Store modules.txt and download new modules not to the installation area, but to a designated location in the users home where write access will be available. The downside of this approach is that this would decrease the security of truly shared installations of Kepler. On the other hand, a problem with malicious modules may be more theoretical than real at this point, as modules are now published and retrieved from a centralized source that we control. Also, if there is a problem with malicious modules, it would affect not just shared installations, but individual installations; protecting shared installations does nothing to protect individual installations which are perhaps just as important. Also, the real risk of malicious modules is low, given current development patterns.

Solution 2: Try to find a way so that Windows installations are not done by default on Windows. This may or may not be possible.

Solution 3: Warn users who do not have appropriate write permissions to the installation area that they may not use the Module Manager. The design of such a warning should be thought through so that it is non-intrusive and graceful. For example, it probably should not pop-up at start-up, but only when the user attempts to invoke Module Manager functionality.

Conclusion: At the very least, users who cannot use the Module Manager due to lack of permission to write to where Kepler has been installed should be warned. Perhaps even better, given the prevalence of this problem in Windows and relatively low security risks, modules.txt and new module downloads should occur in a local area that is writable.

Related issues:

Blocked by Kepler - Bug #5064: Kepler 2.1 Tracking Bug	**New**	**06/30/2010**

History

#1 - 07/29/2010 04:26 PM - David Welker

I think another approach would work better for the problem of permissions on Windows. Moving modules to exist on multiple locations adds serious complications in terms of collisions and limits the ability to work on multiple checkouts. Also, it requires the creation of a registry to track module locations. This is significantly more complicated than storing all modules in the same location relative to the build. This also significantly complicates the IDEs.

A better approach that I am going to research would be to see if Kepler can interact with Windows to log the user in as an administrator if they use the module manager and do not have the appropriate permissions.

#2 - 07/29/2010 05:41 PM - Matt Jones

Running as administrator or other root-level accounts is likely to be a security issue, as it allows escalation of privileges if Kepler is compromised -- or possibly even if it is used as intended to run workflows that, for example, launch command shells. So, I think it best if Kepler only runs as a non-admin user. The easiest solution to this problem is simply to store any files that a user needs to write to in a user-writable directory, probably as a subdirectory of KeplerData in their home directory.

#3 - 07/29/2010 08:13 PM - David Welker

There are some serious complications to the solution of storing modules in multiple locations. This is not a "simple" solution. Let's have a meeting to discuss how to proceed.

#4 - 08/09/2010 09:21 AM - David Welker

To update, this Thursday we had a meeting where, among other things, we discussed how to proceed with this problem. We agreed that one reasonable solution would be for the user to be prompted to enter an administrator password when invoking certain functions of the module manager.

This weekend, I researched the feasibility of this solution on Windows. And this is a non-trivial problem. The feature in Windows that we would need to use is known as User Access Control (UAC). The process of prompting a user for a log on for administrative privileges using that feature is called "elevation." The issue with elevating privileges, is that this generally cannot be done except at program launch. One can either use the "run as" verb when launching a program, or specify that the program needs administrative privileges upon launch.

There are a couple of solutions to move forward here.

(1) Make the module manager an entirely separate program. Selecting the module manager menu item will launch the module manager, but instead of launching it as a dialog in Kepler, it would launch in an entirely different JVM. Upon launch, the module manager would ask for the administrative privileges necessary to modify its install location in Program Files. The issue with this solution is that ending the Kepler process to restart a new Kepler might be very tricky; how would the module manager know what process to shut down if it was running in a different process.

(2) Make the processes that download new modules or change modules.txt separate programs that are invoked in the background. Getting the progress monitor to work in this situation might be tricky, since what would be measured is the progress of another program and not a process occurring within Kepler. Also, if a particular blog entry I have read on this is correct, the separate program may need to be wrapped in another language other than Java. I am doubtful that this is actually correct. For details, check this blog post here:

http://mark.koli.ch/2009/12/uac-prompt-from-java-createprocess-error740-the-requested-operation-requires-elevation.html

I don't see why the separate program that is invoked cannot be written in Java, but the person who wrote this detailed blog post decided to write theirs using Visual C++ presumably based on this belief.

(3) Simply require Windows users to open "Run as Administrator" whenever they want to use the module manager.

(4) Make the module manager a separate program that is not invokable from within Kepler. That separate program would instead always be invoked separately and would automatically prompt for elevated administrative privileges when opened on Windows.

(5) Write modules and modules.txt to a user area. This could lead to significant inefficiencies, as modules would have to be downloaded potentially multiple times on the same computer for different users. Also, such centralization would prevent people from having the same module downloaded and modified differently in different contexts, since reference would now be made to a centralized module. Perhaps a scheme could be made to get around this problem...

Anyway, of these five solutions, I am tentatively leaning towards a variation on solution (5). The reason is that this could be a more universal solution and would not require us to delve into operating specific issues with issues such as with Windows User Access Control. The default is that the build could look for modules as peers to the build-area folder. But a scheme to make them look elsewhere (somewhere in the user directory) for installs could be devised. In this way, the experience for developers would remain unchanged (you could have multiple instances of the same module associated with different builds that you modify differently) but users with installed versions would not have to try to write into Program Files or other protected areas on other operating systems.

Thoughts by others on this problem are welcome, but for now I am going to proceed working out the details on how a solution paralleling (5) could be made to work smoothly.

#5 - 09/01/2010 01:02 AM - David Welker

I have finally implemented a variation on solution (5). Basically, when a flag is present (.utilize.user.kepler.modules) then new modules are downloaded to KeplerData/kepler.modules and the modules.txt from KeplerData/kepler.modules is used. This change required significant changes to the way the build system worked, as well as the module-manager module.

Closing.

#6 - 03/27/2013 02:29 PM - Redmine Admin

Original Bugzilla ID was 5065