**SUMMARY**

A concurrency bug in Eclipse 3.3.2 Platform Resources.

**DETAILS**

Some details can also be found at: [https://bugs.eclipse.org/bugs/show\_bug.cgi?id=226264](https://bugs.eclipse.org/bugs/show_bug.cgi?id=57467)

This bug is due to a data race.

1. Job (Thread 1) has the org.eclipse.core.internal.resources.WorkManager.lock and does a Project.delete() which calls Resource.delete().In that method the WorkManager.lock is temporarily released and deleteUnprotected() is executed w/o that lock.

2. The second Job (Thread 2) was waiting to acquire the Workmanager.lock and can obtain it now. In its Workspace.endOperation() it starts to broadcast changes and NotificationManager.notify() calls Workspace.setTreeLocked() (so its field Workspace.treeLocked contains now thread 2).

3. In the meantime in Thread 1 runs and does also a broadcast of the Project deletion, this is also done in the NotificationManager.notify() and it also calls setTreeLocked(), effectively setting the field Workspace.treeLocked to thread 1.

4. Now in Thread 2 all calls to Workspace.isTreeLocked() give the \_wrong\_ result false.

|  |  |
| --- | --- |
| Thread1 () | Thread2 () |
| public void setTreeLocked(boolean locked) {  treeLocked = locked ? Thread.currentThread() : null;  } | public boolean isTreeLocked() {  return treeLocked == Thread.currentThread();  } |
| 0-lock | 0-lock |