TaskManager notes

Classes That Reference TaskManager

Class	Package	Notes
AbortJob	com.sun.jini.mahalo	Subclass of Job. Passed a TaskManager as parameter. Uses ParticipantTask, no dependencies.
CommitJob	com.sun.jini.mahalo	Subclass of Job. Passed a TaskManager as parameter. Uses ParticipantTask, no dependencies.
EventType	com.sun.jini.norm.event	Task type SendTask, subclass of RetryTask, no dependencies.
EventTypeGenerator	com.sun.jini.norm.event	Supplies a TaskManager for use by the EventType objects it generates.
FiddlerImpl	com.sun.jini.fiddler	Extensive use of TaskManager, with many different Task subtypes. No dependencies.
Job	com.sun.jini.mahalo	Manage performance of a job as a set of tasks all of which need to be created by the Job subclass. There is some dubious code in performWork that silently throws away an exception that would indicate internal inconsistency.
JoinManager	net.jini.lookup	Uses ProxyRegTask, which extends RetryTask. Special problem - making sure a service gets exactly one ID. If the ID has already been allocated, no dependencies. If not, runAfter any ProxyRegTask with lower sequence number, ensuring that only the lowest sequence number ProxyRegTask in the TaskManager can run. Safe if, and only if, tasks are submitted in sequence number order.
LeaseRenewalManager	net.jini.lease	Uses QueuerTask and RenewTask. No dependencies.
LookupDiscovery	net.jini.discovery	Uses DecodeAnnouncementTask and UnicastDiscoveryTask. No dependencies.
LookupLocatorDiscovery	net.jini.discovery	Uses DiscoveryTask. No dependencies.
MailboxImpl	com.sun.jini.mercury	Uses a NotifyTask, subclass of RetryTask, no dependencies.
Notifier	com.sun.jini.outrigger	Uses its own NotifyTask, subclass of RetryTask. Dependency based on EventSender runAfter test. EventSender has two implementations. An EventRegistrationWatcher.BasicEventSender waits for any BasicEventSender belonging to the same EventRegistrationWatcher. VisibilityEventSender has no dependencies.

ParticipantTask	com.sun.jini.mahalo	No dependencies.
PrepareAndCommitJob	com.sun.jini.mahalo	Subclass of Job. Passed a TaskManager as parameter. Uses ParticipantTask, no dependencies.
PrepareJob	com.sun.jini.mahalo	Subclass of Job. Passed a TaskManager as parameter. Uses ParticipantTask, no dependencies.
RegistrarImpl	com.sun.jini.reggie	Uses multiple Task types: AddressTask - no dependencies; DecodeRequestTask - no dependencies; EventTask - run after EventTask for same listener, "Keep events going to the same listener ordered"; SocketTask - no dependencies.
RetryTask	com.sun.jini.thread	Abstract class implementing Task. It provides for automatic retry of failed attempts, where an attempt is a call to tryOnce.
ServiceDiscoveryManager	net.jini.lookup	Uses CacheTask - no dependencies; ServiceIdTask - run after ServiceIdTask with same ServiceId and lower sequence number. Its subclasses NewOldServiceTask and UnmapProxyTask inherit runAfter. ServiceIdTask's subclass NotifyEventTask runs after RegisterListenerTask or LookupTask with same ProxyReg and lower sequence, and also calls the ServiceId runAfter. Bug ID 6291851. Comment suggests the writer thought it was necessary to do a sequence number check to find the queue order: " and if those tasks were queued prior to this task (have lower sequence numbers)".
SettlerTask	com.sun.jini.mahalo	Subclass of RetryTask. No dependencies. Used in TxnManagerImpl.
TxnManagerImpl	com.sun.jini.mahalo	Uses SettlerTask and ParticipantTask. No dependencies.
TxnManagerTransaction	com.sun.jini.mahalo	Creates a TaskManager, threadpool, and passes it around to e.g. Job and AbortJob.
TxnMonitor	com.sun.jini.outrigger	Uses TxnMonitorTask.
TxnMonitorTask	com.sun.jini.outrigger	Subclass of RetryTask. No dependencies.

Issues

RetryTask

RetryTask is a Task implementation whose run method tries a subclass supplied method with a boolean result. If the method returns false, indicating failure, the RetryTask's run method schedules another try in the future, using a WakeupManager supplied to the RetryTask constructor.

During the time between a failed attempt and its retry, there does not seem to be any control to prevent conflicting tasks from entering the same TaskManager. Some of those tasks would have waited for the task being retried, if it had been in the TaskManager at their time of arrival. Delayed retry and dependence on sequence number seem incompatible. Notifier.NotifyTask and JoinManager.ProxyRegTask both extend RetryTask and have dependencies. JoinManager.ProxyRegTask uses a sequence number, but problem does not need to, and should not. The intent seems to be to run tasks for a given service one-at-a-time until its ServiceId has been set.

ServiceDiscoveryManager.CacheTask

Most subclasses inherit a "return false;" runAfter. The exceptions are ServiceIdTask, its subclasses, and LookupTask. Both have sequence number dependencies. It is not yet clear whether ServiceDiscoveryManager is ensuring that tasks enter the TaskManager in sequence number order. If it does, the code is correct, but wastes time with a trivially true check. If not, the code is incorrect relative to the comments, which seem to expect order.