Flux and Quartz

A Features Comparison

Reading Time: 8 minutes

We are often asked to provide a comparison of Flux to Quartz as part of a due diligence from a prospect's development management. The question implicitly asked is "Why pay for something like Flux when open-source and free projects like Quartz provide what we need?" The answer, simply put, is that the Quartz functionality is a very small subset of the functionality present in Flux. Beyond that, the question needed to be explored is "how much are you willing to spend in development, test, and support to create a solution that best fits your needs?"

Note that this document is not intended in any way to dismiss Quartz. Quartz is a capable and valuable component in many applications. Many organizations and applications utilize Quartz. But some organizations have found that as their applications evolve a more complete solution becomes essential. They see a need to move from simple job scheduling to complex workflow orchestration. See "*It's a Simple Matter of Programming. Really?*" at http://flux.ly/resources/smop.pdf for a discussion of this.

A Quick Overview

All the functionality present in Quartz is present in Flux, so Flux is a true superset of Quartz. Flux was developed before Quartz. Flux is very actively maintained and supported. Flux has a long history of feature additions. Flux has evolved into a file orchestration platform, supporting workflow, scheduling, design, security, and monitoring features necessary for enterprise processing. For example whereas Quartz has a small number of triggers, Flux has many including timer triggers, database condition triggers, mail processing triggers, web service triggers, file exist/not exist triggers and the ability to add custom triggers and extend existing triggers.

Quartz provides for calling Java code using their listener interface to perform actions. Flux provides this feature too, and Flux includes out-of-the-box shell and batch script actions, database actions, mail actions, web service actions, file operation and transfer actions, compression/decompression, and encryption/decryption actions. All integrated and supported into a cohesive product.

Quartz (the open source version) does not provide a graphical user interface, integrated security, audit logging, an integrated repository of workflow components, a monitoring dashboard, and numerous other features that Flux provides out of the box. Quartz does not provide control over where jobs are run.

Flux Provides Beyond Quartz

Flux provides a powerful runtime variable substitution capability that significantly increases the degree of reuse of workflows. The same workflow can have very different processing characteristics depending upon the namespace of the workflow.

Flux also provides integrated security features to secure the Flux execution engine with communications using SSL and all operations requiring authentication and appropriate authorization.

Other key Flux features include:

- Ability to pin the running of specific workflow group to specific servers
- Ability to set concurrency throttles on specific workflow groups
- Agents to distribute and delegate work across the nodes in a distributed network
- Managed file transfer with real time monitoring of the transfer

Finally, Flux is a supported commercial offering with email and telephone support staffed by experienced software engineers. All versions of Flux in production today (some over 10 years old) are still supported.

Feature Comparison

Quartz Features list below copied from http://quartz-scheduler.org/overview/features as of 12/5/2014 and edited to fit this document.

Quartz Features	Flux Features
• Current Release is 2.2.1 as of 24 Sept 2013	• Current Release is 8.0.9 as of 27 Oct 2014. 8.0.10 is scheduled for release 1st quarter of 2015. • Flux generally releases a new update, with new features and improvements, every 3-4 months.
Runtime Environments	
• Quartz can run embedded within another free standing application	• Yes
• Quartz can be instantiated within an application server (or servlet container), and participate in XA	• Yes

transactions	
• Quartz can run as a stand-alone program (within its own Java Virtual Machine), to be used via RMI	• Yes – and Flux also provides command line utilities and a REST API. Flux has retired its RMI interface dues to its lack of security and network limitations.
• Quartz can be instantiated as a cluster of stand-alone programs (with load-balance and fail-over capabilities) for the execution of jobs	• Yes
Job Scheduling	
• Jobs are scheduled to run when a given Trigger occurs. Triggers can be created with nearly any combination of the following directives:	• Yes- and Flux provides many other ways to trigger workflows besides time based events. The existence of files, the non-existence of files, and results of a database query, incoming REST/SOA calls, and incoming mails are some of the additional triggers provided by Flux.
• at a certain time of day (to the millisecond)	• Yes
• on certain days of the week	• Yes
• on certain days of the month	• Yes
• on certain days of the year	• Yes
• not on certain days listed within a registered Calendar (such as business holidays)	• Yes
• repeated a specific number of times	• Yes
• repeated until a specific time/date	• Yes
• repeated indefinitely	• Yes
• repeated with a delay interval	• Yes
• Jobs are given names by their creator and can also be organized into named groups.	• Yes
• Triggers may also be given names and placed into groups, in order to easily organize them within the scheduler. Jobs can be added to the scheduler once, but registered with multiple Triggers.	• Yes – triggers are given names within groups (i.e., namespaces) within Flux. Flux allows multiple triggers to fire a workflow.
• Within an enterprise Java environment, Jobs can perform their work as part of a distributed (XA) transaction.	• Yes

Job Execution	
• Jobs can be any Java class that implements the simple Job interface, leaving infinite possibilities for the work your Jobs can perform.	• Yes
• Job class instances can be instantiated by Quartz, or by your application's framework.	• Yes.
• When a Trigger occurs, the scheduler notifies zero or more Java objects implementing the JobListener and TriggerListener interfaces (listeners can be simple Java objects, or EJBs, or JMS publishers, etc.). These listeners are also notified after the Job has executed.	• Yes – and Flux also provides actions that allow one to call existing Java classes dynamically without having to use the Flux listener interface.
• As Jobs are completed, they return a JobCompletionCode which informs the scheduler of success or failure. The JobCompletionCode can also instruct the scheduler of any actions it should take based on the success/fail code - such as immediate re-execution of the Job.	• Yes – and Flux provides that ability for actions to pass much more than completion codes. Complex Java objects containing the results of the job can be returned. In addition, a Flux workflow can start other Flux workflows based on the results of prior jobs. • Flux also provides timeout and error flows to direct the workflow in exceptional circumstances.
Job Persistence	
• The design of Quartz includes a JobStore interface that can be implemented to provide various mechanisms for the storage of jobs.	• Yes – Flux supports this feature although it is implemented differently. The storage of workflows, and workflow templates, are provided out of the box without any required 'implementation.'
• With the use of the included JDBCJobStore, all Jobs and Triggers configured as "non-volatile" are stored in a relational database via JDBC.	• Yes – Flux supports this feature although it is implemented differently. Flux provides the ability to finely control the points in a workflow that are committed to the relational database.
• With the use of the included RAMJobStore, all Jobs and Triggers are stored in RAM and therefore do not persist between program executions - but this has the advantage of not requiring an external database.	• Yes – and Flux supports this via the use of an in-memory database. No changes are required to the Flux workflows themselves other than a configuration setting.

Transactions	
• Quartz can participate in JTA transactions, via the use of JobStoreCMT (a subclass of JDBCJobStore).	• Yes – Flux supports this feature although it is implemented differently.
• Quartz can manage JTA transactions (begin and commit them) around the execution of a Job, so that the work performed by the Job automatically happens within a JTA transaction.	• Yes
Clustering	
• Fail-over.	• Yes
• Load balancing.	• Yes
• Quartz's built-in clustering features rely upon database persistence via JDBCJobStore (described above).	• Yes – Flux supports this feature although it is implemented differently.
• Terracotta extensions to Quartz provide clustering capabilities without the need for a backing database.	• Yes - Flux supports this using an in-memory database implementation
Listeners & Plug-Ins	
• Applications can catch scheduling events to monitor or control job/trigger behavior by implementing one or more listener interfaces.	• Yes
• The Plug-In mechanism can be used add functionality to Quartz, such keeping a history of job executions, or loading job and trigger definitions from a file.	• Yes – and this is provided by Flux without needing to develop
• Quartz ships with a number of "factory built" plug-ins and listeners.	• Yes
Recoverability and Idempotence	
• In-progress Jobs marked "recoverable" are automatically re-executed after a scheduler fails. This means some of the job's "work" will be executed twice.	• Yes – and Flux provides fine-grained control over where a workflow restarts in the event the workflow fails. Flux allows custom error handlers to be constructed and assigned to names spaces to create consistent recovery paths.
• This means the job should be coded in such a way that its work is idempotent.	• Flux does not require that workflows be coded as idempotent due to its finer grain control and more flexible error and exception handling.

Selecting Where Jobs Run	
• **Terracotta Quartz Where** (not included in open source versions of Quartz) is a feature that provides customers the ability to control where jobs execute — based on machine name (user-defined node or node group) or machine resources (RAM, CPU or OS)	• Yes
• The Plug-In mechanism can be used add functionality to Quartz, such keeping a history of job executions, or loading job and trigger definitions from a file.	• Yes – and Flux provides the ability to keep a history of job executions, or loading job and trigger definitions from a file out of the box with no required plug-ins.
• Quartz ships with a number of "factory built" plug-ins and listeners.	• Yes

In Summary

In summary, comparing Quartz and Flux is a bit like comparing a single tool to a factory. Flux integrates many capabilities into a robust platform dedicated to making short work of an enterprise's workflow scheduling and file processing.

For additional information regarding Flux review the Flux documentation at http://doc.flux.ly, the Java API at http://support.flux.ly/80/javadoc/ and the REST API at http://support.flux.ly/80/restapi/.

About Flux

Flux assists enterprises in provisioning, onboarding, scheduling, tracking, and reporting an enterprise's file orchestration processes. These orchestrations vary from simple file transfers to highly complex workflows involving extensive processing, many routes, varied alerts, and complex decisioning. First released in 2000, Flux has grown into a file orchestration and workflow scheduling platform that enterprises rely on daily for their mission critical systems.

Contact Flux

Contact sales at: sales@flux.ly
For further information browse: flux.ly