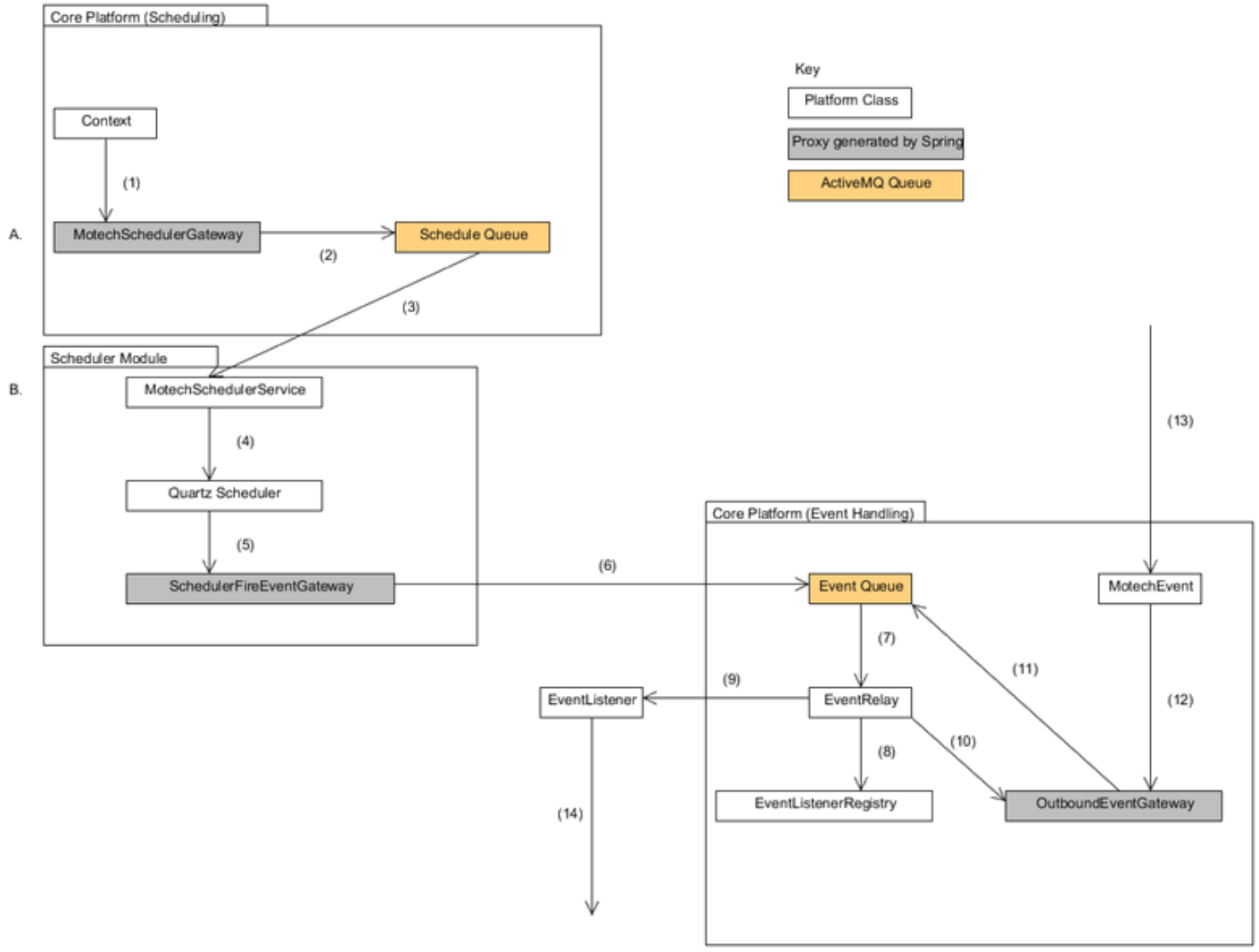
MOTECH - Mobile Technology for Community Health

Event handling and scheduling architecture

There are currently two ways to schedule a job in MOTECH. The first (A., see below) uses the MotechSchedulerGateway, found in a global Context object defined in the server module, to pass scheduled jobs as JMS messages to a schedule queue. The queue will dequeue these events and the implementor must provide an inbound channel to accept and route the jobs to the scheduler of their choice. The second method (B., see below) invokes the MotechSchedulerService found in the Scheduler module directly. The service in this case is injected into the class that uses it. This service employs Quartz to schedule MotechScheduledJobs. When triggered, MotechScheduledJobs raise events that are sent to an event queue. These messages are dequeued and then received by a consumer, the event relay, which in turn discovers all of the listeners on the event and invokes the appropriate method that was listening on that event.



Notes

1. In order to use the MotechSchedulerGateway, the implementor must retrieve it from the Motech Context bean defined in applicationPlatformServer.xml (in the motech-platform-server-api module).

2. The MotechSchedulerGateway is generated by Spring at runtime. It is configured in outboundScheduleJobChannelAdapter.xml (in the motech-platform-server-api module). JMS messages are sent to an active MQ queue, defined in the activemq.properties file (in the motech-platform-server module).

3. The schedule queue will dequeue the messages and send them to consumers. The core platform does not have any consumer for the schedule queue. Implementors would have to define these consumers unless they use the scheduler module. The scheduler module provides a MotechSchedulerService that receives these messages. The configuration file schedulerInboundChannelAdapter.xml (found in motech-platform-scheduler) specifies an inbound channel adapter that routes the payload, based on its type, to the scheduler service. Currently, unscheduleJob from the MotechSchedulerGateway is unsupported, and the service provides several methods not found in the gateway.

4. The scheduler service retreives the Quartz Scheduler from the SchedulerFactoryBean. The service can only schedule MotechScheduledJobs, which all must include a MotechEvent.

5. When the trigger for the scheduled job is satisfied, the job is executed and the accompanying Motech event is sent to the SchedulerFireEventGateway interface. This gateway is defined in schedulerFiredEventChannelAdapter.xml and a proxy is generated by Spring at runtime.

6. The SchedulerFireEventGateway serves as an outbound message gateway to shuttle messages to an adapter that then sends JMS messages to the event queue.

7. The event queue dequeues messages and routes them to its consumers. The core platform has one consumer for events, a channel that routes messages to an EventRelay interface (implemented by ServerEventRelay) which accepts the dequeued MotechEvent objects. This is configured in inboundEventChannelAdapter.xml (found in motech-platform-server-api). ServerEventRelay's relayEvent method is not a method found in its interface, EventRelay.

8. The event relay retreives the listeners that are listening on the motech event it is relaying (listening is based on the value bound to the motech event's subject key).

9. If the event has a specific destination or only one listener, the listener's handle method is called. This will invoke the method that was annotated as a MotechListener. These listener classes can be project specific and will reside outside of the core platform.

10. If the event has multiple listeners, a separate event is raised for each listener and sent to the outbound event gateway. The original event object is not handled by a listener in this case.

11. The outbound event gateway, defined in outboundEventChannelAdapter.xml (motech-platform-common), routes the JMS message to the event queue. Each message is then dequeued and sent to the consumers, as in step 7.

12. and **13.** The OutboundEventGateway, which is responsible for shuttling events to the event queue, may receive project specific events from outside of the platform. These are all handled identical to steps **7-11.**

14. The EventListener's handle method will invoke the method that was annotated as a MotechListener. The MotechEvent will be passed to that method as a parameter.

UML of interfaces involved in the schedule/event handling of the core platform