TTM
Total Transaction Management

AMon

A Monitoring System for ActiveMQ

Joe Fernandez
joe.fernandez@ttmsolutions.com
Total Transaction Management, LLC
570 Rancheros Drive, Suite 140
San Marcos, CA 92069
760-591-0273
www.ttmsolutions.com

AMon

- Designed specifically for monitoring the ActiveMQ (AMQ) message broker process; it cannot be used to directly monitor AMQ clients or other queuing systems.
- Comprises two primary modules: Monlet Container and SNMP Agent. Both are embedded within the AMQ message broker process.
- The Monlet container hosts custom-built monitoring agents (Monlets), and facilitates the rapid development of the Monlets.
- A Monlet comprises conditional expressions and their corresponding actions; the Monlet invokes actions when the corresponding conditions are satisfied.
- The Monlet container gives its Monlets direct access to the message broker's runtime objects (i.e., overall runtime state) and also to some of the JVM's runtime objects.
- A variable of a conditional expression represents a particular runtime object (e.g., queue or topic) in the AMQ message broker.

AMon

TTM
Total Transaction Management

- Examples of Monlets and the objects they monitor.
    - Resource Monlet – tests whether a particular runtime object (e.g., queue, topic, etc.), within either the message broker or JVM, has breached a threshold. The action may be to either invoke a script, fire off an email, or raise an SNMP notification (trap).
    - Audit Monlet – tests whether a message that is being routed through the broker meets a condition of some kind (e.g., includes some user-defined property that has been assigned a particular value). The action may be to write the contents of the message out to an audit log.
    - Exception Monlet – captures exceptions thrown by the message broker. The action may be to send an email containing the contents of the stack trace.

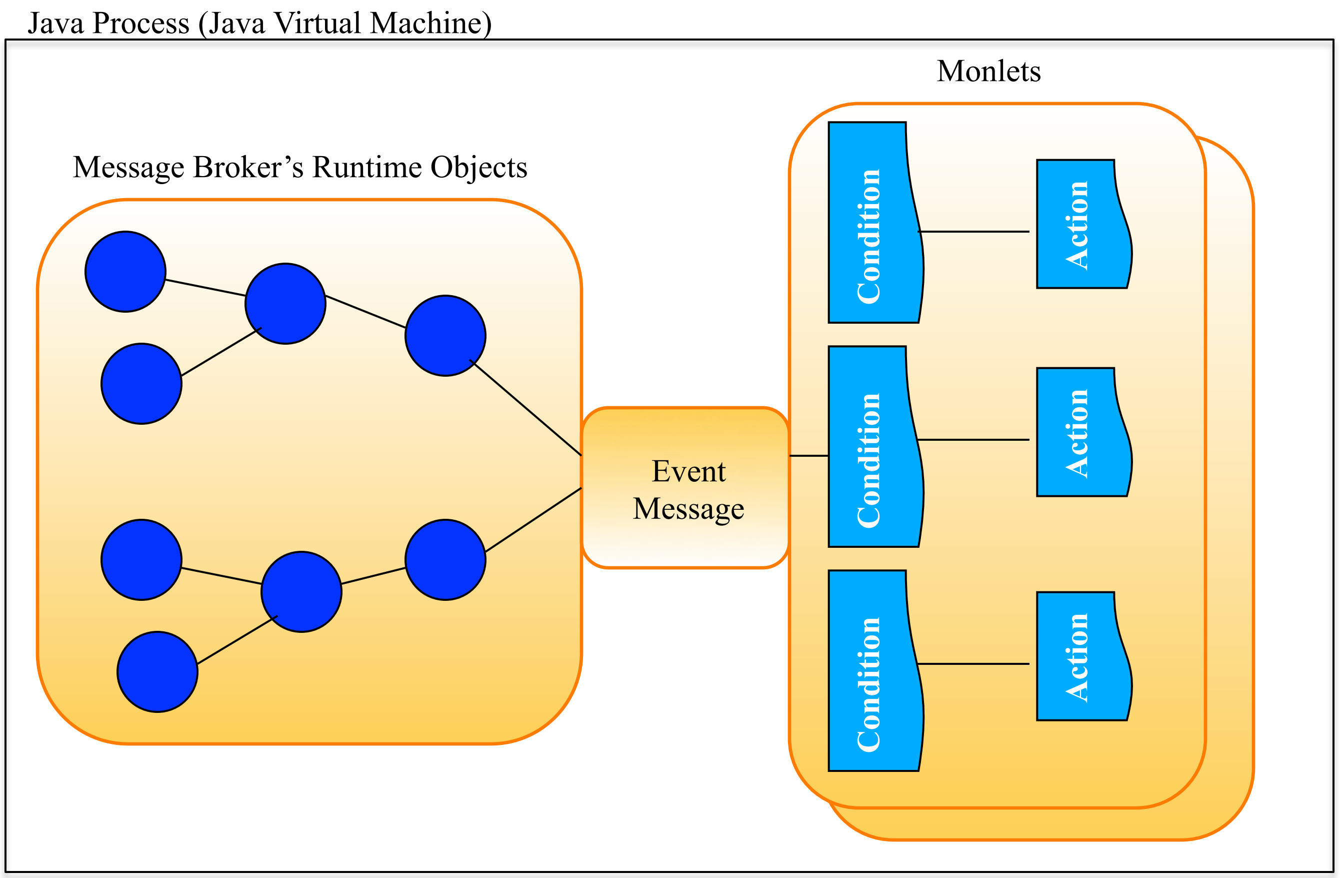
AMon

TTM
Total Transaction Management

- Monlet container maintains a clean separation between Monlets and the broker/ JVM that they are monitoring. This separation is provided by the following:
  - Event messages published by the Monlet framework and consumed by Monlets.
  - AMon's monitoring expression language (MEL)
  - Apache Camel integration framework and its DSL.
- The Monlet container is installed as an AMQ message broker plugin; therefore, it is included in AMQ's main processing event chain.
- Inclusion in the event chain allows the container to publish event messages that correspond to each of AMQ's broker processing events.
- The container is configured to publish messages that pertain to all or any subset of these processing events.
- The container also generates an asynchronous 'Timer' event and 'AMQ Exception' event.

AMon

- A Monlet subscribes to all or any subset of event message types.
- An event message contains object references to the message broker's runtime objects. So the message acts as a type of portal into the broker's runtime objects.



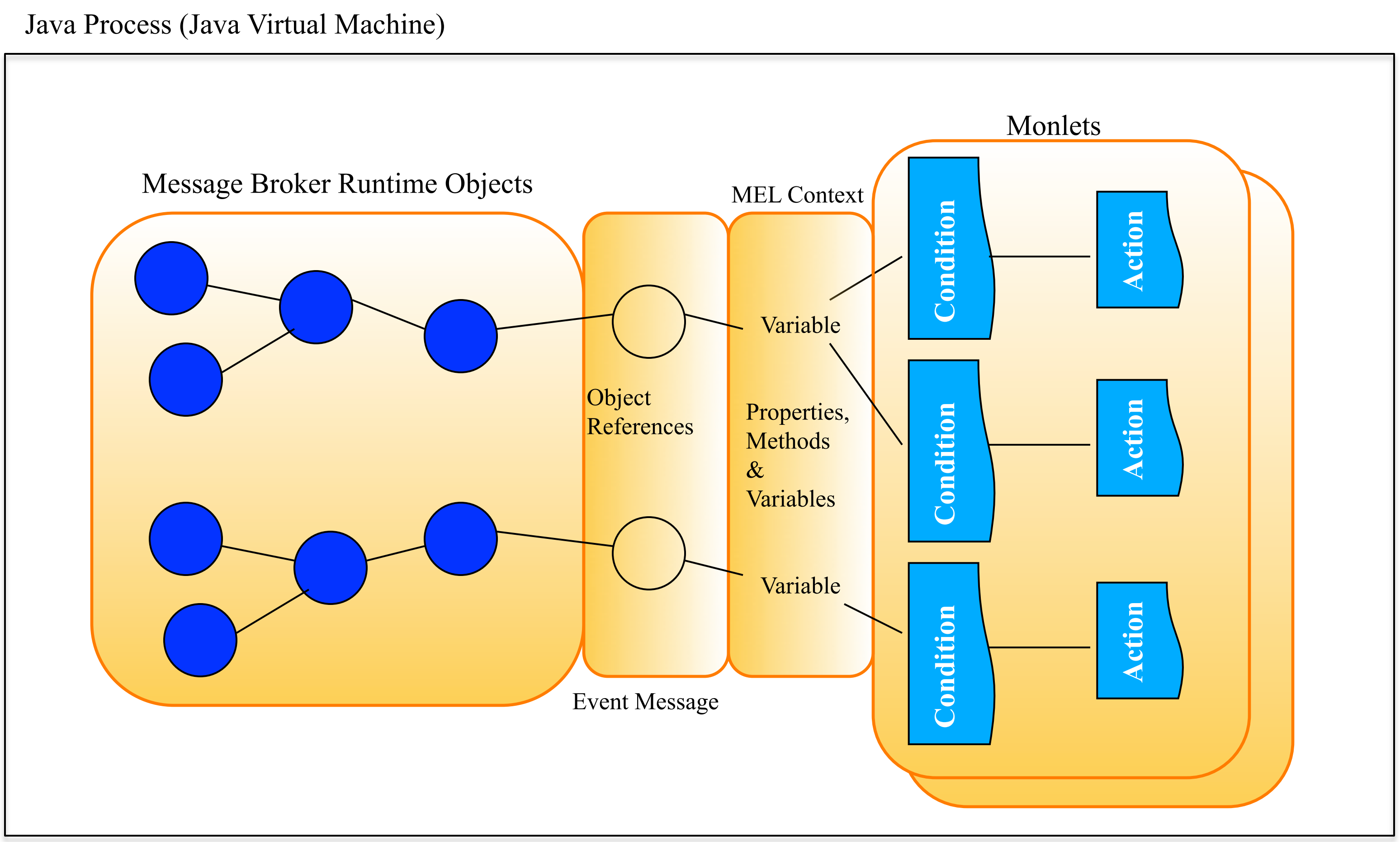
Java Process (Java Virtual Machine)
Monlets
Message Broker's Runtime Objects
Event Message
Condition
Action
Condition
Action
Condition
Action

AMon

- AMon's monitoring expression language (MEL) is an extension of the Java Unified Expression Language (JUEL).
- All of AMQ's runtime objects adhere to the JavaBeans standard. The MEL leverages that standard to access those objects' properties.
- AMon creates a MEL context for an event message. The context includes the binding of MEL variables to the message broker objects, as well as MEL-specific methods/functions and properties.
- Simple example of a MEL conditional expression: ${destination.name = = 'TEST.Q'}

AMon

Total Transaction Management



AMon

TTM
Total Transaction Management

- Monlets are implemented as Apache Camel routes.
- Camel is a powerful Spring-based integration framework that is used to implement the enterprise integration patterns defined in Gregor Hohpe and Bobby Woolf's book titled, "*Enterprise Integration Patterns*".
- One of the unique features of Camel is that it offers a Java Domain Specific Language (DSL) that is used by Camel end-users to:
  - Quickly implement all kinds of messaging patterns (routes).
  - Interface to many different endpoints/components that are inherently supported by Camel; e.g., SMTP, TCP/IP, FTP, JMS, HTTP, Servlets, AWS, AMQP, etc. A very long list! You can also add your own custom components.
- Camel's Java DSL is combined with the MEL to facilitate the quick development of Monlets.

AMon

- AMon includes a SNMP v2 Agent and accompanying Management Information Base (MIB)
- The combination of these two components allow ActiveMQ to be monitored by SNMP-capable management systems.
- The SNMP Agent allows Monlets to generate SNMP traps as an action
- The MIB lists and describes all the ActiveMQ attributes that can be monitored via AMon's SNMP agent
- The MIB attributes are a reflection of ActiveMQ's JMX MBeans and their attributes

AMon

- AMon and Monlets are configured through an external properties file. Through this properties file, you can:
  - Control the event message types that AMon will publish. By default, publishing of all event messages is disabled.
  - Define Monlet-specific properties.
  - Define SNMP-specific properties
- Updating the properties can occur *dynamically*; i.e., there is no need to restart the message broker if and when you update the properties.
- AMon adheres to the JMX, so it can be managed/administered through any JMX client. Through a JMX-client you can
  - Start and stop AMon
  - Update and persist configuration properties

AMon

TTM
Total Transaction Management

If you have any questions, please contact me:

joe.fernandez@ttmsolutions.com