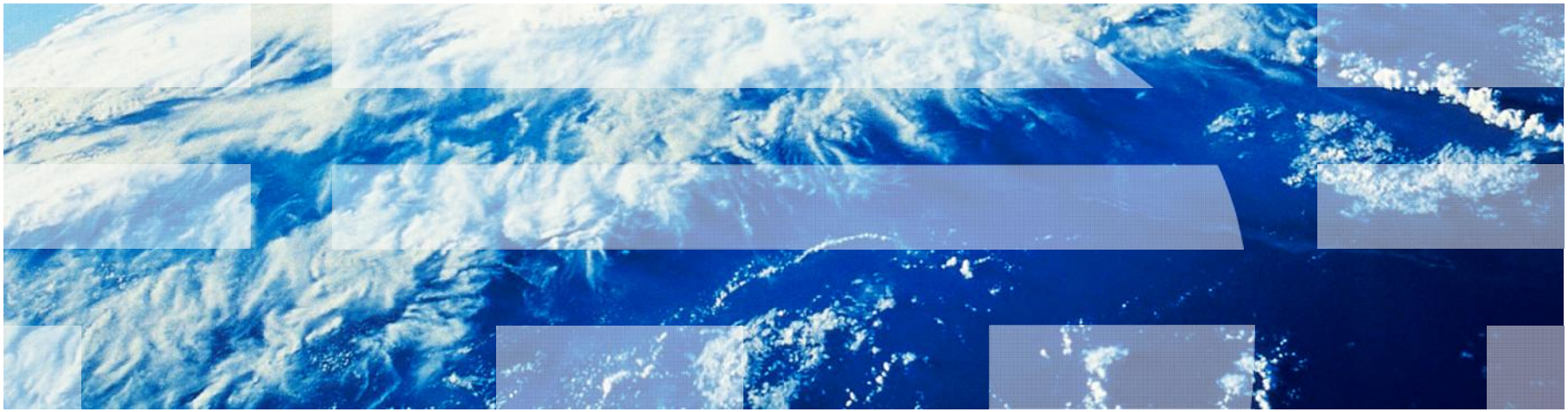
Sandeep Khare (Sandeep.Khare@in.ibm.com)

08th October 2013

IBM Sterling Agent Architecture



This session will be recorded and a replay will be available on IBM.COM sites and possibly social media sites such as YouTube. When speaking, do not state any confidential information, your name, company name or any information that you do not want shared publicly in the replay. By speaking during this presentation, you assume liability for your comments.

Agenda

- Introduction
- Agent – Key Terms
- Flow of the agent
- Scenarios – how agent can be used to scale up
- Agent failure recovery mechanism
- Advantages of Agent

Introduction

- IBM Sterling Agent is a time triggered transaction that performs a variety of individual functions. It is not triggered by conditions, events or user input.
- Agent can be configured to trigger automatically at a specific time interval or it can be triggered manually.
- JMS queue plays an important part in functioning of the agent as these are used by the agent to read and write messages.
- IBM Sterling Agent is particularly useful to automate processing of thousands of records automatically.

Key Terms

- **getJobs()**

  getJobs() is a method which fetches the records which agent has to process and puts these records in the JMS queue. These messages are picked up by the executeJobs() method. getJobs() method reads getJobs (trigger) message.

- **executeJobs()**

  executeJobs() is a method which picks record from JMS queue and processes them one record at a time. Records in JMS queue are posted as execute message by the getJobs() method.

- **Number of records to Buffer**

  Number of records to Buffer is batch size in which getJobs() method will fetch the records.

- **No. of Threads**

  Number of concurrent threads with which Agent runs

- **Trigger Message**

  Trigger message is a signal to the agent to start processing. This message will have details about the agent. Based on this details agent will retrieve the first set of job.

  Following is the trigger message posted to JMS queue for scheduleOrder Agent

```
<Message FlowName="schedule" TransactionKey="SCHEDULE.0001">
 <AgentDetails>
    <MessageXml Action="Get" CollectPendingJobs="Y"
       MaximumRecords="5" NumRecordsToBuffer="10"/>
 </AgentDetails>
</Message>
```

- **getJobs message**

  This message is posted to the JMS queue which getJobs method reads and fetches the next set of records. Next set of records is determined by the LastMessage element and NumRecordsToBuffer i.e. if NumRecordsToBuffer is 10 then 10 records will be fetched by the getJobs method from the last key which was processed earlier. In first run of getJobs method it will fetch first 10 records.

- Following is the subsequent getJobs message posted to JMS queue for scheduleOrder Agent

```
<Message FlowName="schedule" TransactionKey="SCHEDULE.0001">
   <AgentDetails>
      <MessageXml Action="Get" CollectPendingJobs="Y"
         MaximumRecords="5" NumRecordsToBuffer="10"/>
   </AgentDetails>
   <LastMessage LastMessagesAdded="10">
      <TaskQueue TaskQKey="2011062002272149586">
         <TransactionFilters Action="Get" CollectPendingJobs="Y"
            DocumentParamsKey="0001" DocumentType="0001"
               MaximumRecords="5" NumRecordsToBuffer="10"
            ProcessType="ORDER_FULFILLMENT"
            ProcessTypeKey="ORDER_FULFILLMENT"
            TransactionId="SCHEDULE.0001" TransactionKey="SCHEDULE.0001"/>
      </TaskQueue>
   </LastMessage>
</Message>
```

getJobs message posted to JMS queue is similar to trigger message except the LastMessage element. Trigger message is often referred to as first getJobs message.

- **execute message**

  This message is posted to JMS queue by getJobs() method. executeJobs() method picks this message and processes the record. This message will have detail about one record which needs to be processed. There will be one execute message posted to JMS queue by getJobs() method for each record fetched by it.

Following is the execute message posted to JMS queue for scheduleOrder Agent

```
<Message FlowName="schedule" TransactionKey="SCHEDULE.0001">
  <AgentDetails>
    <TaskQueue TaskQKey="2011062002272149586">
      <TransactionFilters Action="Get" CollectPendingJobs="Y"
        DocumentParamsKey="0001" DocumentType="0001"
        MaximumRecords="5" NumRecordsToBuffer="10"
        ProcessType="ORDER_FULFILLMENT"
        ProcessTypeKey="ORDER_FULFILLMENT"
        TransactionId="SCHEDULE.0001" TransactionKey="SCHEDULE.0001"/>
    </TaskQueue>
  </AgentDetails>
</Message>
```

- **Trigger Agent**

  Trigger agent is process which sends trigger message manually to agent if agent is not auto trigger

- **Agent Server Name**

  Server name on which agent will run. Multiple agents can have same server name.

- **JMS Queue Name**

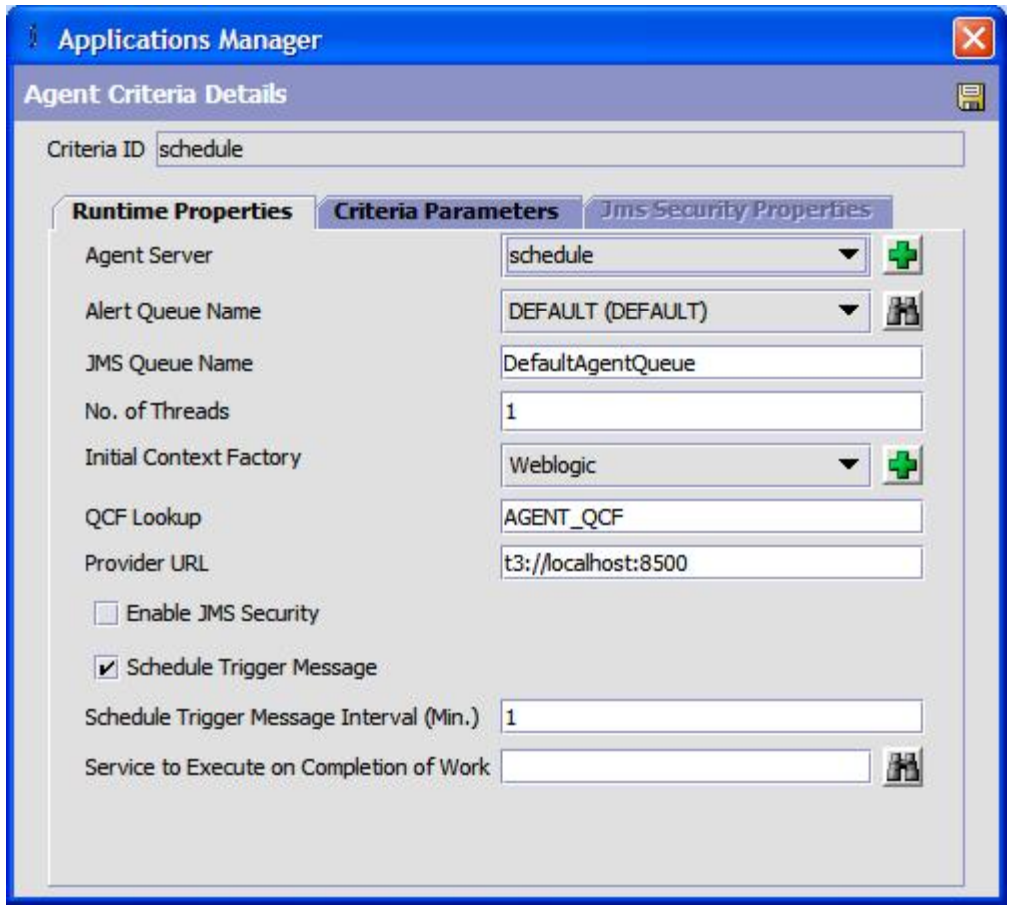
  JMS Queue name with which Agent will interact (put and read messages from JMS queue)

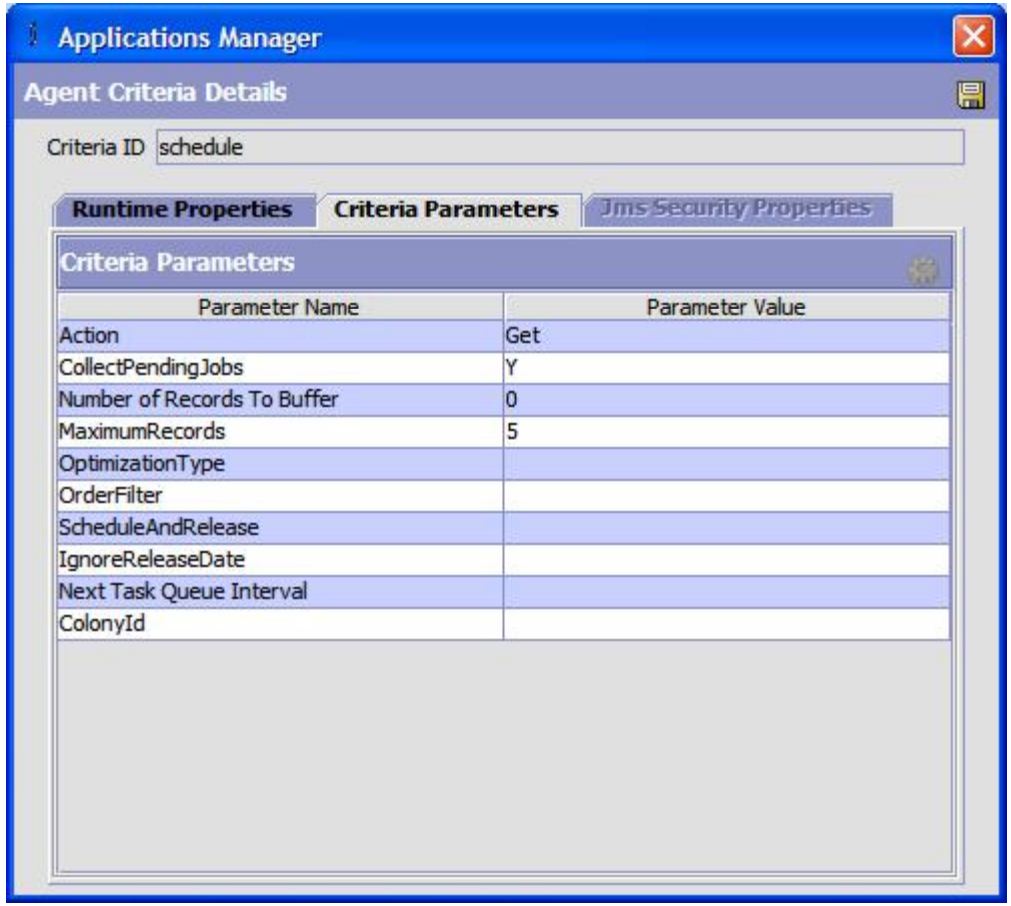
- **QCF Lookup**

  The name of the JMS queue connection factory

- **Initial Context Factory**

  JMS queue type used. Valid values are Websphere (MQSeries), Weblogic (weblogic JMS), Jboss (JBOSS JMS), File(MQSeries with file) and TIBCO.





Starting the Agent

- Agent can be started by running agentserver.sh script from %INSTALL_DIR%/bin folder.
- agentserver.sh script takes agent server name as the argument.

  %INSTALL_DIR%/bin/agentserver.sh ServerName
- After executing above command, all the agents configured with agent server name as ServerName will be started.

Agent Trigger

Agent can be triggered by following two methods

1. Manually: For manual trigger, triggeragent script has to be invoked manually

2. Automatic: For automatic trigger, schedule trigger message has to be checked and time interval needs to be configured in agent configuration.

   Trigger agent script (triggeragent.sh or triggeragent.cmd) can be found in INSTALL_DIR%/bin folder.

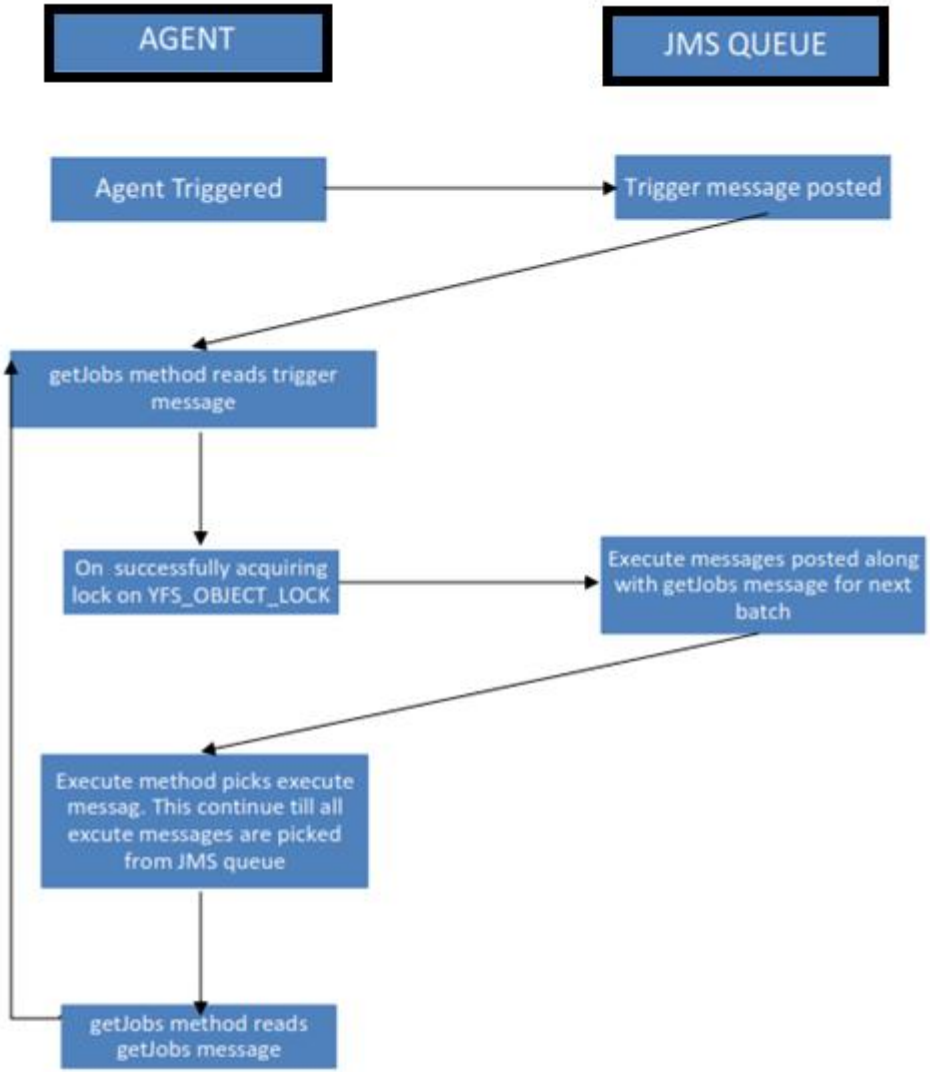
   %INSTALL_DIR% is the folder location where Sterling is installed.

   Agent trigger puts the **first** getJobs message into JMS queue. This message is picked up by getJobs() method

   Agent will not be triggered again till all the present records are processed i.e. while agent is processing records then manual trigger will not be honored. Auto trigger will not happen till all the records are processed.

Flow of the Agent

- Agent is started
- Agent is triggered (manually through triggeragent.sh or automatically)
- getJobs (trigger) message is posted to JMS queue by agent trigger
- getJobs() method reads the above message
- Within getJobs() method, agent tries to acquire lock on YFS_OBJECT_LOCK table for agent Criteria ID
- If lock is not available then getJobs() method exits and does nothing.
- If lock is available then getJobs() method fetches records which needs to be processed.
- Above records are posted as execute message to JMS queue
- After the execute messages, one getJobs message is also posted with last record key.
- execute method picks execute message one by one and processes them.
- After all the execute messages are consumed then only getJobs message is left in queue
- getJobs() method picks up the getJobs message left in the queue.



JMS Queue Picture

This is how JMS queue looks during various stages of processing . Agent is running with Number of records to buffer=10.

1. Before Agent is starts

   JMS Queue => empty

2. Agent is triggered

   JMS Queue => T

3. Trigger message is picked up by getJobs method. getJobs method posts execute message along with getJobs message

   JMS Queue => E1, E2, E3, E4, E5, E6, E7, E8, E9, E10, G1

4. After all the execute message is picked and processed by the executeJobs method.

   JMS Queue => G1

5. G1 is picked up by the getJobs method to retrieve next set of records. These records are posted as execute message along with getjobs message for next batch

6. JMS Queue => E11, E12, E13, E14, E15, E16, E17, E18, E19, E20, G2

T	Trigger Message
E	Execute Message
G	getJobs message

Single Thread versus Multiple thread

Agent can be run in single thread mode (No. of Threads=1) or in multi thread mode (No. of Threads>1). Flow of agent explained previously remains same. In single thread mode, after the getJobs method has put message into JMS queue then executeJobs method will pick up one message at time, process record present in message and then pick the next message.

In multi thread mode, after the getJobs method has put message into JMS queue then there are multiple instances of executeJobs method. Number of instances of executeJobs method is equal to No. of Threads configured for agent. These instances of executeJobs works in parallel without interfering each other. Each executeJobs method will pick up one message from JMS queue and process the record present in message and then pick up the next available message. getJobs method does not run in parallel mode as executeJobs method.

Running Agent in multiple JVMs

One Agent can be run across multiple JVMs (different command windows). This is done for agent which has huge number of records to process and a single JVM is reaching its memory/processing limitation with the number of threads configured for agent. This feature helps agent scale up to meet up the number of records to process.

For example, if an agent is configured to run with five threads and this agent is running on two JVMs then ten threads of this agent is running. To make sure getJobs method is not running at same point in two JVMs, agent checks for the lock on YFS_OBJECT_LOCK table. getJobs method which is able to acquire the lock will go and fetch the records for agent. getJobs method which fails to acquire lock will exit without doing anything.

YFS_OBJECT_LOCK is table where lock for particular agent is maintained.

Running multiple Agents on same JVM

Multiple Agents can be run on one single JVM. This can be done to club the multiple Agents which have lesser load.

For this all Agents should have same server name in agent configuration.

Failure recovery mechanism

What happens if agent shutdowns in between while processing records.

- If JMS queue is persistent queue.

  JMS queue is persistent queue, messages in JMS queue will not be lost. When agent starts again, it will start reading and processing the messages from the JMS queue.

- If JMS queue is non persistent queue

  If messages which were present in the JMS queue when agent went down are present in JMS queue then it will start reading and processing the messages from the JMS queue when it starts again.

  If messages which were present in JMS queue when agent went down are lost then agent will start afresh.

Custom Agent

- Can be written by implementing out of the box interface (YCPBaseTaskAgent or YCPBaseAgent). Refer to core java docs for more details on these interface.
- Custom logic needs to be implemented by overriding getJobs and executeJobs method.
- Architecture for Custom Agent is same as out of the box agent's architecture

Integration Agent

- Integration Agent runs asynchronous service (configured in Application Manager).
- Integration Agent's flow is driven by the asynchronous service it is running. It does not follow flow or architecture of the agent.

Advantages of the Agent

- Failure recovery mechanism
- Runs automatically in background with minimal human interference
- Processes large volume of data
- Multi threaded
- Supports multiple JVMs

Thank You!