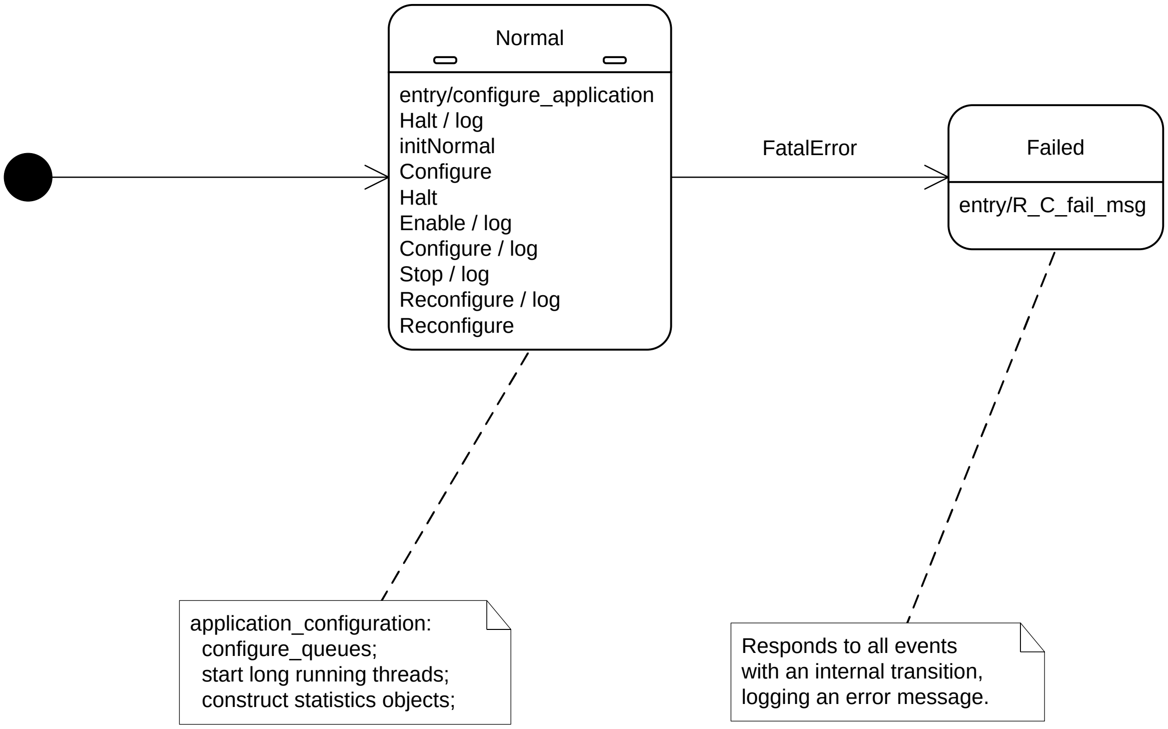
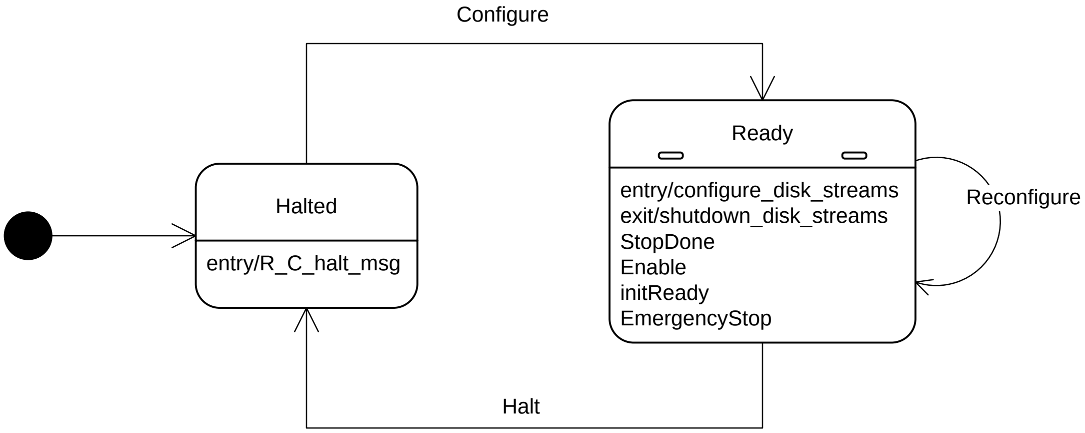
Top-level states



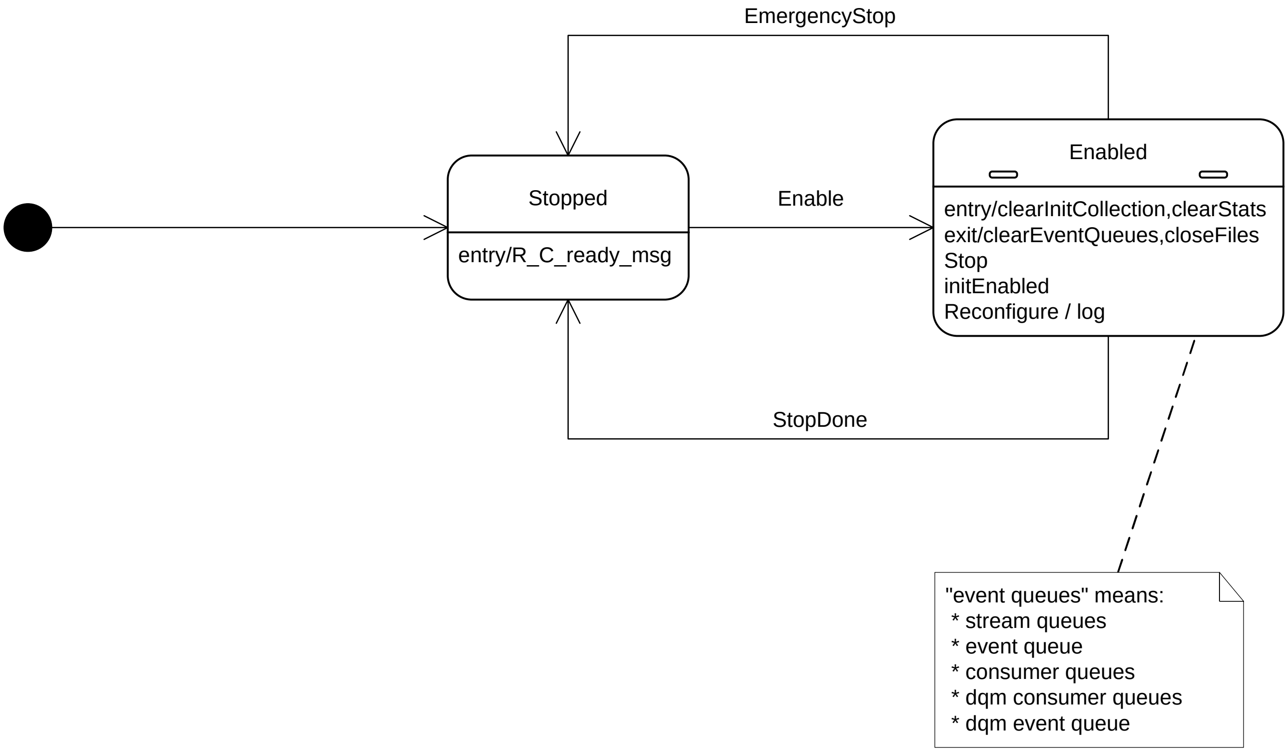
application_configuration:
  configure_queues;
  start long running threads;
  construct statistics objects;

Responds to all events
with an internal transition,
logging an error message.

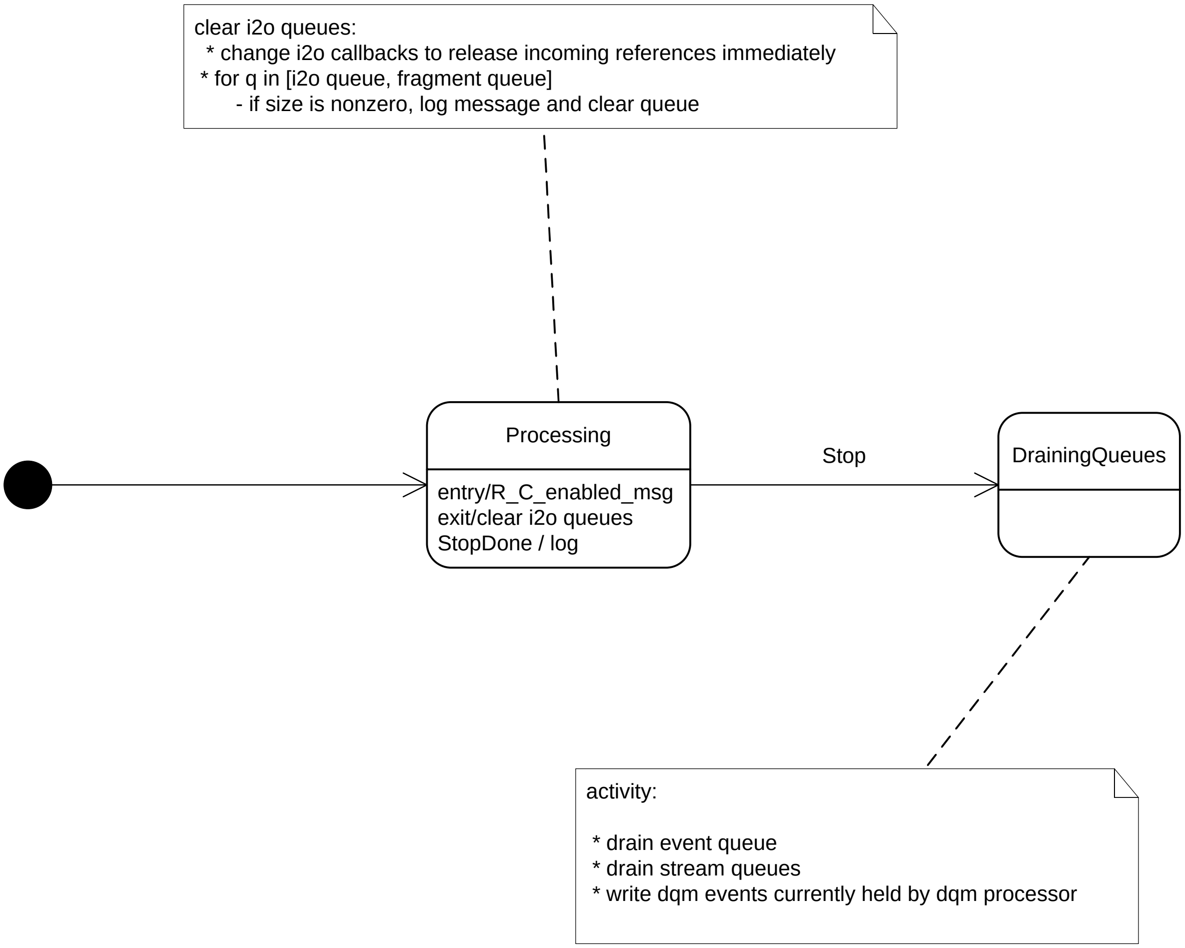
Substates for *Normal* state.



Substates for *Ready* state

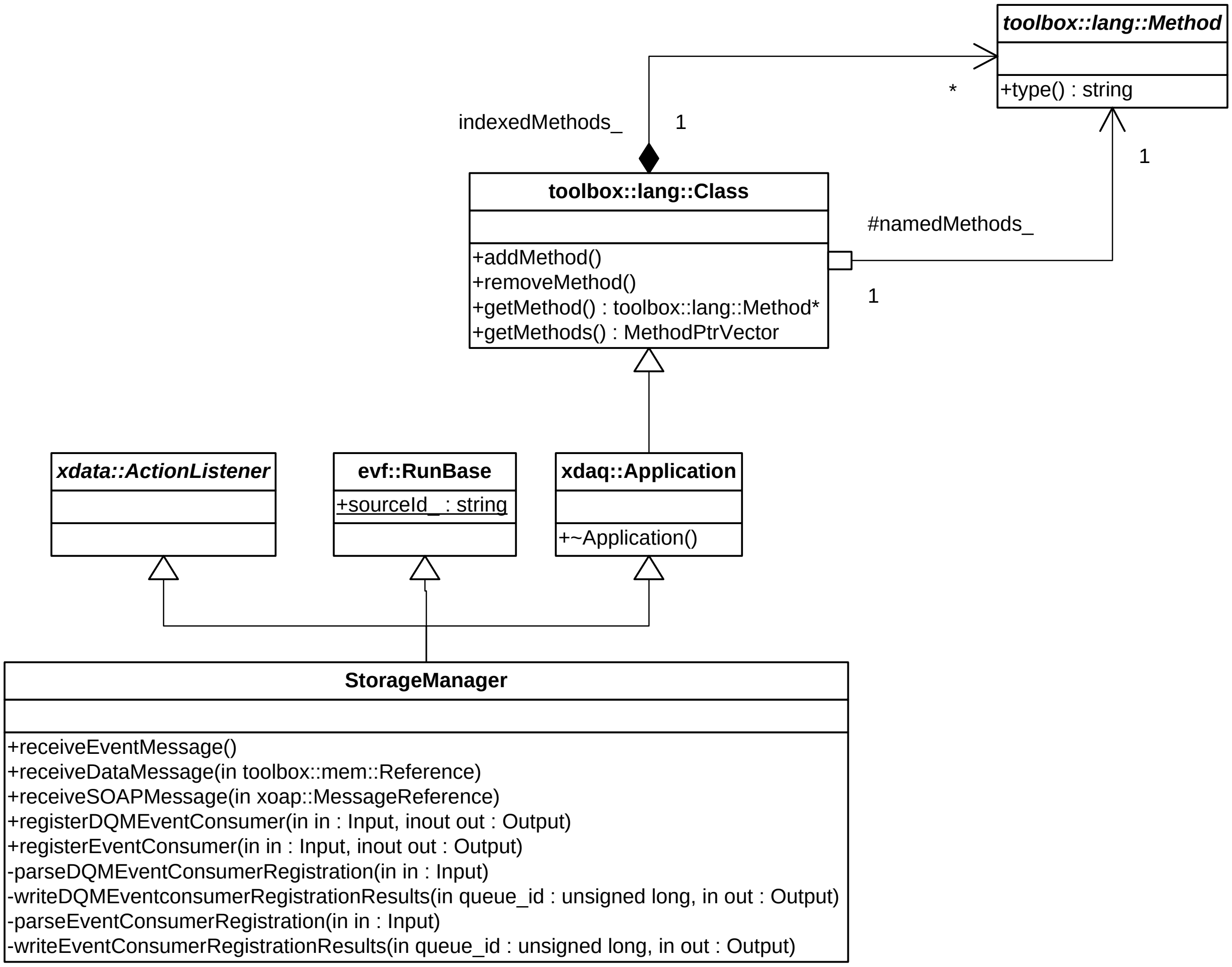


Substates for *Enabled* State

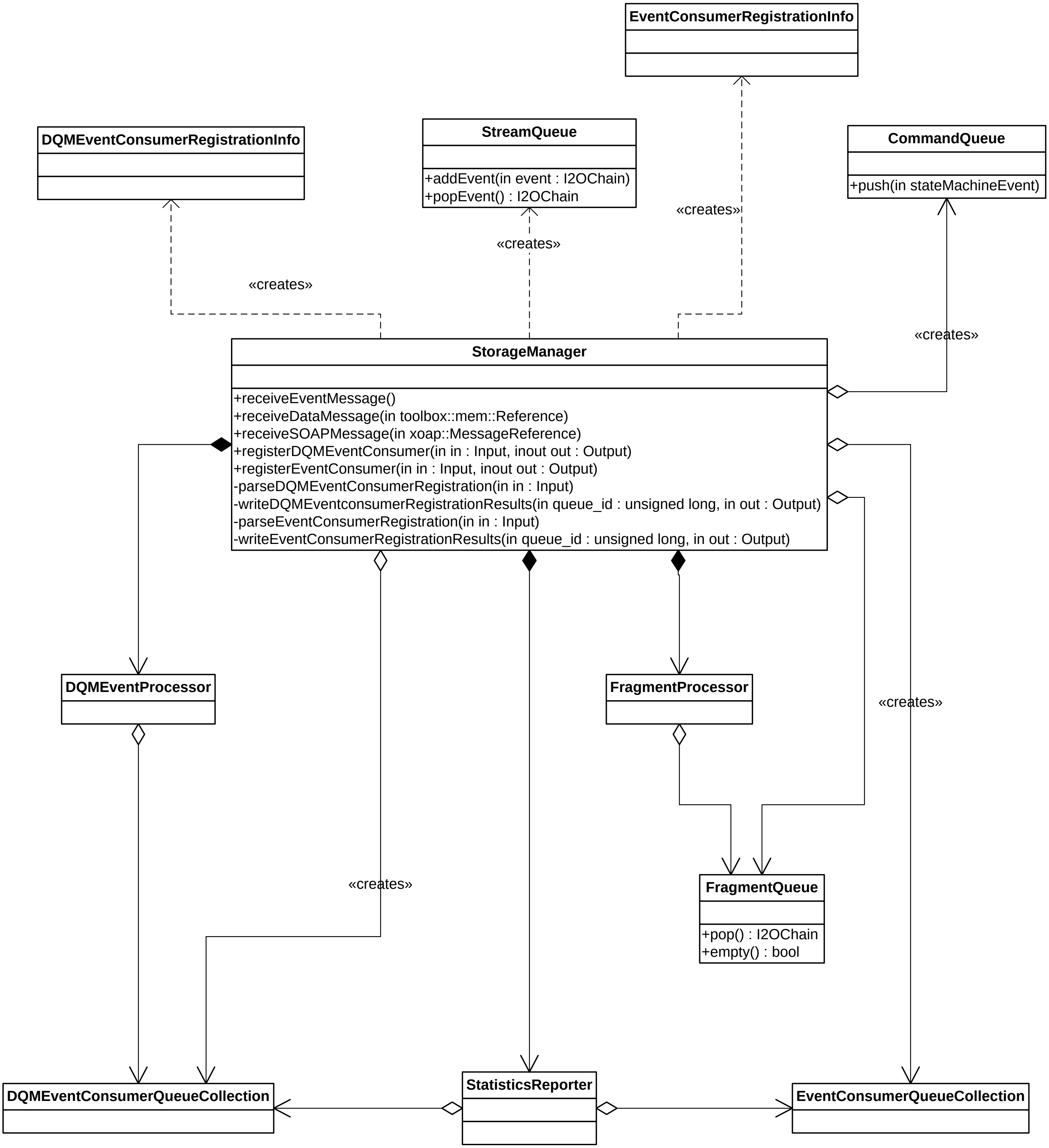


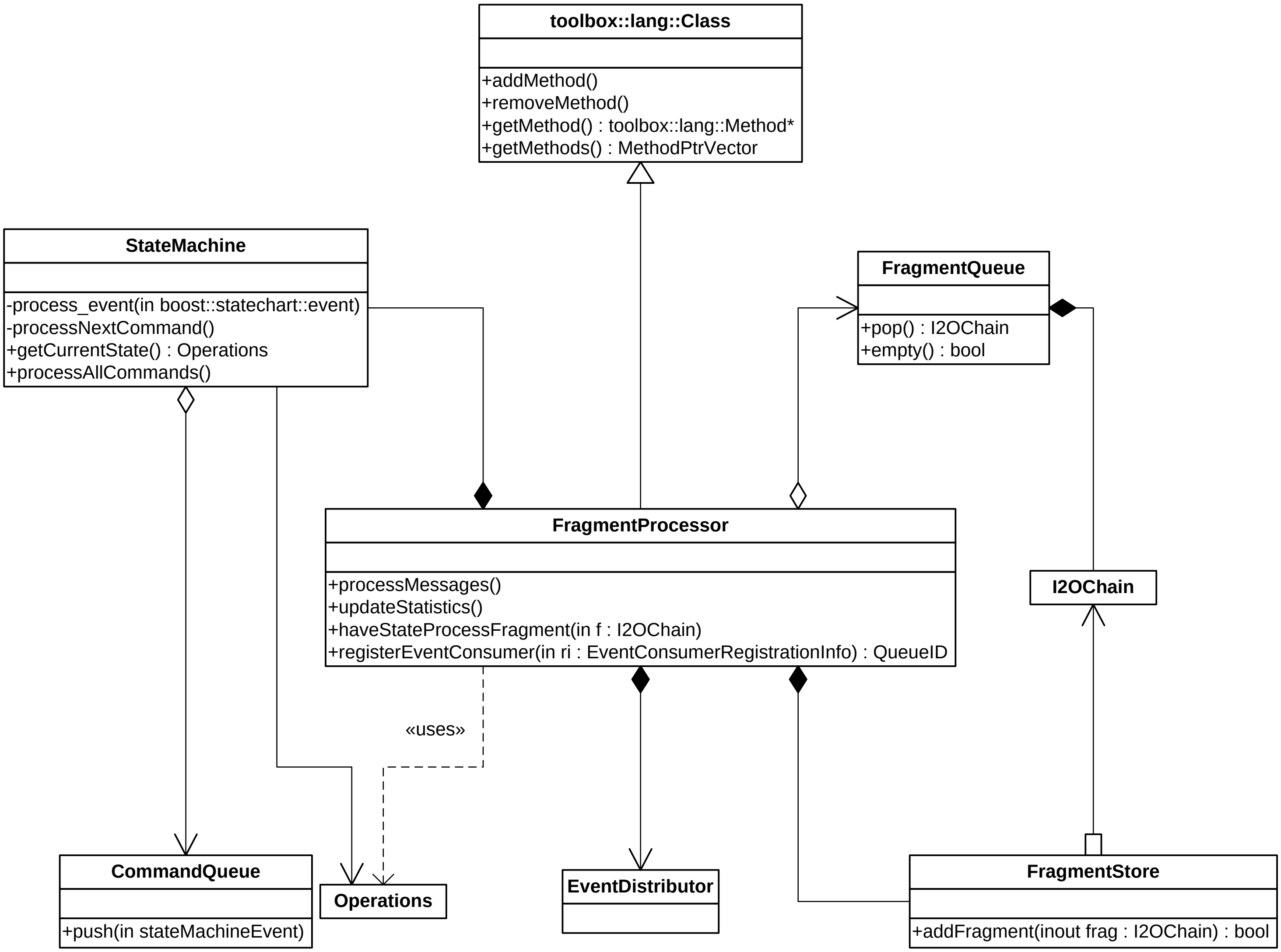
```
clear i2o queues:
  * change i2o callbacks to release incoming references immediately
 * for q in [i2o queue, fragment queue]
       - if size is nonzero, log message and clear queue
```

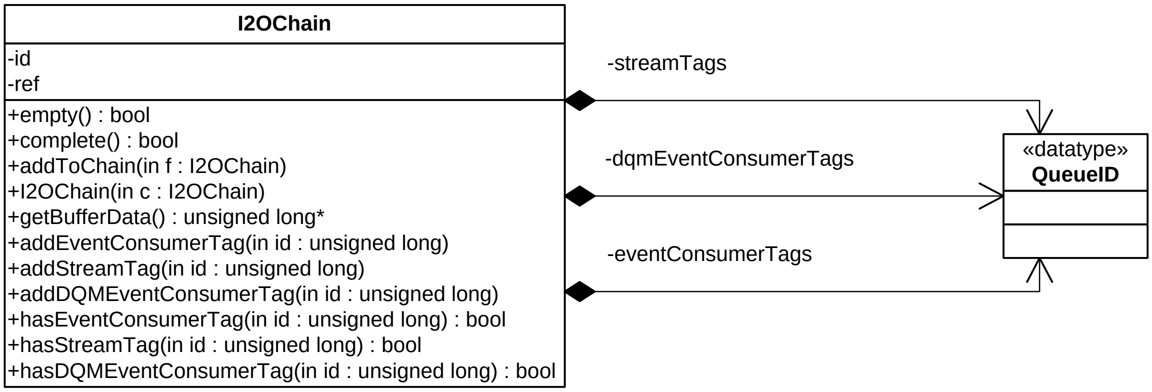
```
activity:

 * drain event queue
 * drain stream queues
 * write dqm events currently held by dqm processor
```

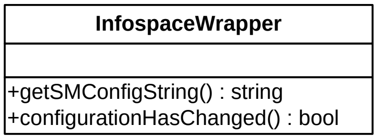


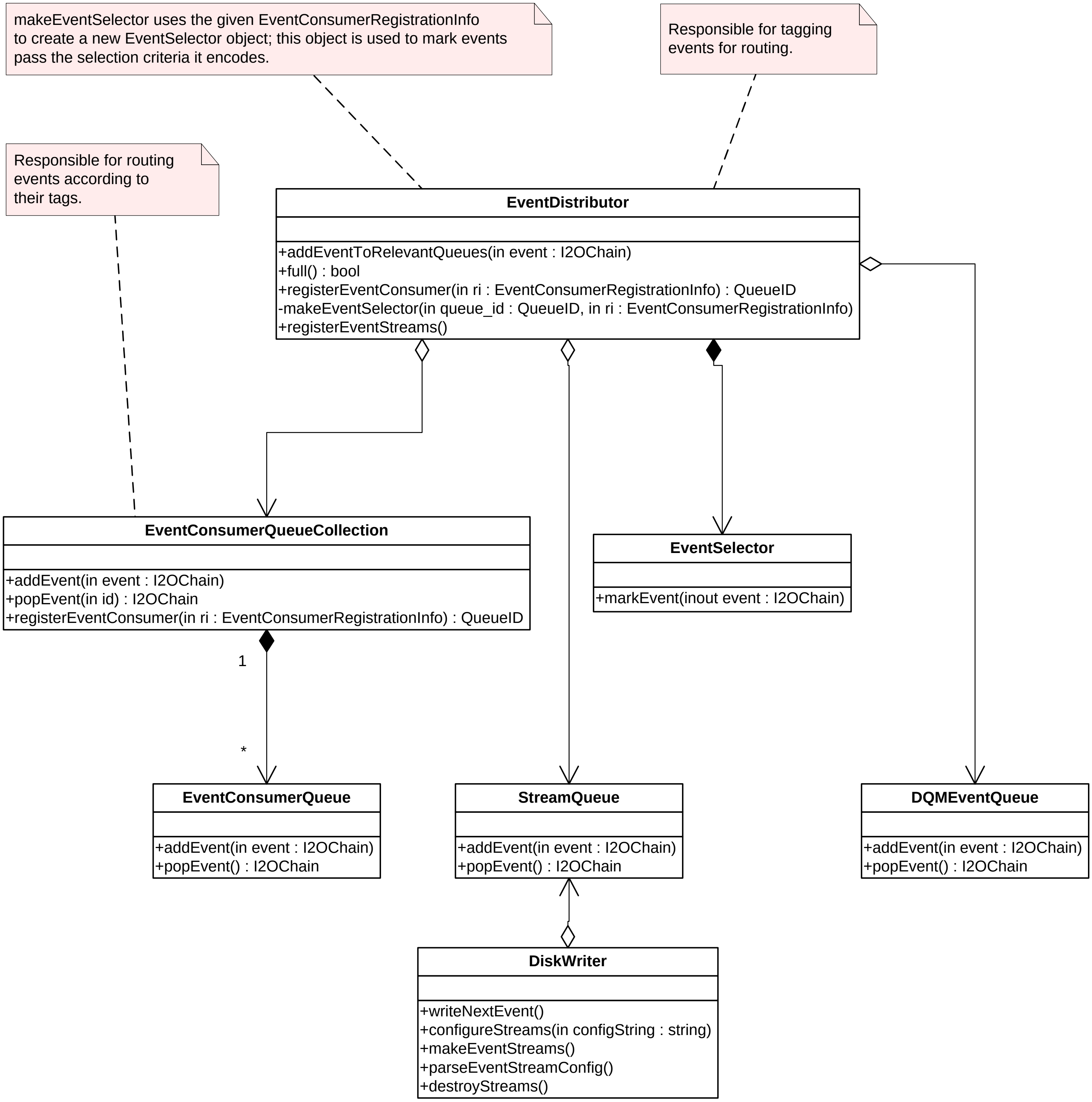
toolbox::lang::Method
+type() : string
*
indexedMethods_
1
1
toolbox::lang::Class
+addMethod()
+removeMethod()
+getMethod() : toolbox::lang::Method*
+getMethods() : MethodPtrVector
#namedMethods_
1
xdata::ActionListener
evf::RunBase
+sourceId_ : string
xdaq::Application
+~Application()
StorageManager
+receiveEventMessage()
+receiveDataMessage(in toolbox::mem::Reference)
+receiveSOAPMessage(in xoap::MessageReference)
+registerDQMEventConsumer(in in : Input, inout out : Output)
+registerEventConsumer(in in : Input, inout out : Output)
-parseDQMEventConsumerRegistration(in in : Input)
-writeDQMEventconsumerRegistrationResults(in queue_id : unsigned long, in out : Output)
-parseEventConsumerRegistration(in in : Input)
-writeEventConsumerRegistrationResults(in queue_id : unsigned long, in out : Output)


EventConsumerRegistrationInfo
DQMEventConsumerRegistrationInfo
StreamQueue
+addEvent(in event : I2OChain)
+popEvent() : I2OChain
CommandQueue
+push(in stateMachineEvent)
«creates»
«creates»
«creates»
«creates»
StorageManager
+receiveEventMessage()
+receiveDataMessage(in toolbox::mem::Reference)
+receiveSOAPMessage(in xoap::MessageReference)
+registerDQMEventConsumer(in in : Input, inout out : Output)
+registerEventConsumer(in in : Input, inout out : Output)
-parseDQMEventConsumerRegistration(in in : Input)
-writeDQMEventconsumerRegistrationResults(in queue_id : unsigned long, in out : Output)
-parseEventConsumerRegistration(in in : Input)
-writeEventConsumerRegistrationResults(in queue_id : unsigned long, in out : Output)
DQMEventProcessor
FragmentProcessor
«creates»
FragmentQueue
+pop() : I2OChain
+empty() : bool
«creates»
DQMEventConsumerQueueCollection
StatisticsReporter
EventConsumerQueueCollection


toolbox::lang::Class
+addMethod()
+removeMethod()
+getMethod() : toolbox::lang::Method*
+getMethods() : MethodPtrVector
StateMachine
-process_event(in boost::statechart::event)
-processNextCommand()
+getCurrentState() : Operations
+processAllCommands()
FragmentQueue
+pop() : I2OChain
+empty() : bool
FragmentProcessor
+processMessages()
+updateStatistics()
+haveStateProcessFragment(in f : I2OChain)
+registerEventConsumer(in ri : EventConsumerRegistrationInfo) : QueueID
I2OChain
«uses»
CommandQueue
+push(in stateMachineEvent)
Operations
EventDistributor
FragmentStore
+addFragment(inout frag : I2OChain) : bool


I2OChain
-id
-ref
+empty() : bool
+complete() : bool
+addToChain(in f : I2OChain)
+I2OChain(in c : I2OChain)
+getBufferData() : unsigned long*
+addEventConsumerTag(in id : unsigned long)
+addStreamTag(in id : unsigned long)
+addDQMEventConsumerTag(in id : unsigned long)
+hasEventConsumerTag(in id : unsigned long) : bool
+hasStreamTag(in id : unsigned long) : bool
+hasDQMEventConsumerTag(in id : unsigned long) : bool
-streamTags
-dqmEventConsumerTags
-eventConsumerTags
«datatype»
QueueID


InfospaceWrapper
+getSMConfigString() : string
+configurationHasChanged() : bool

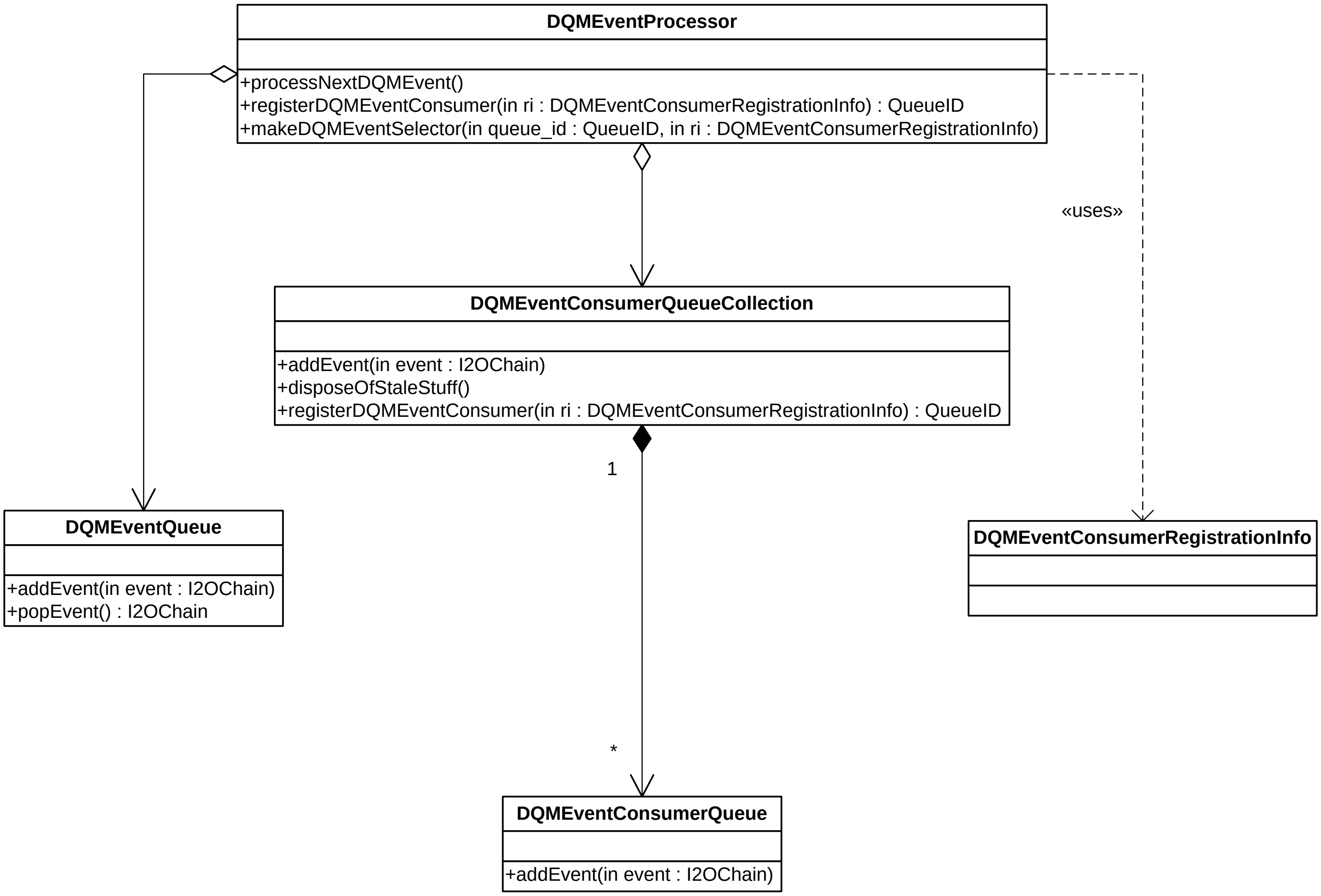


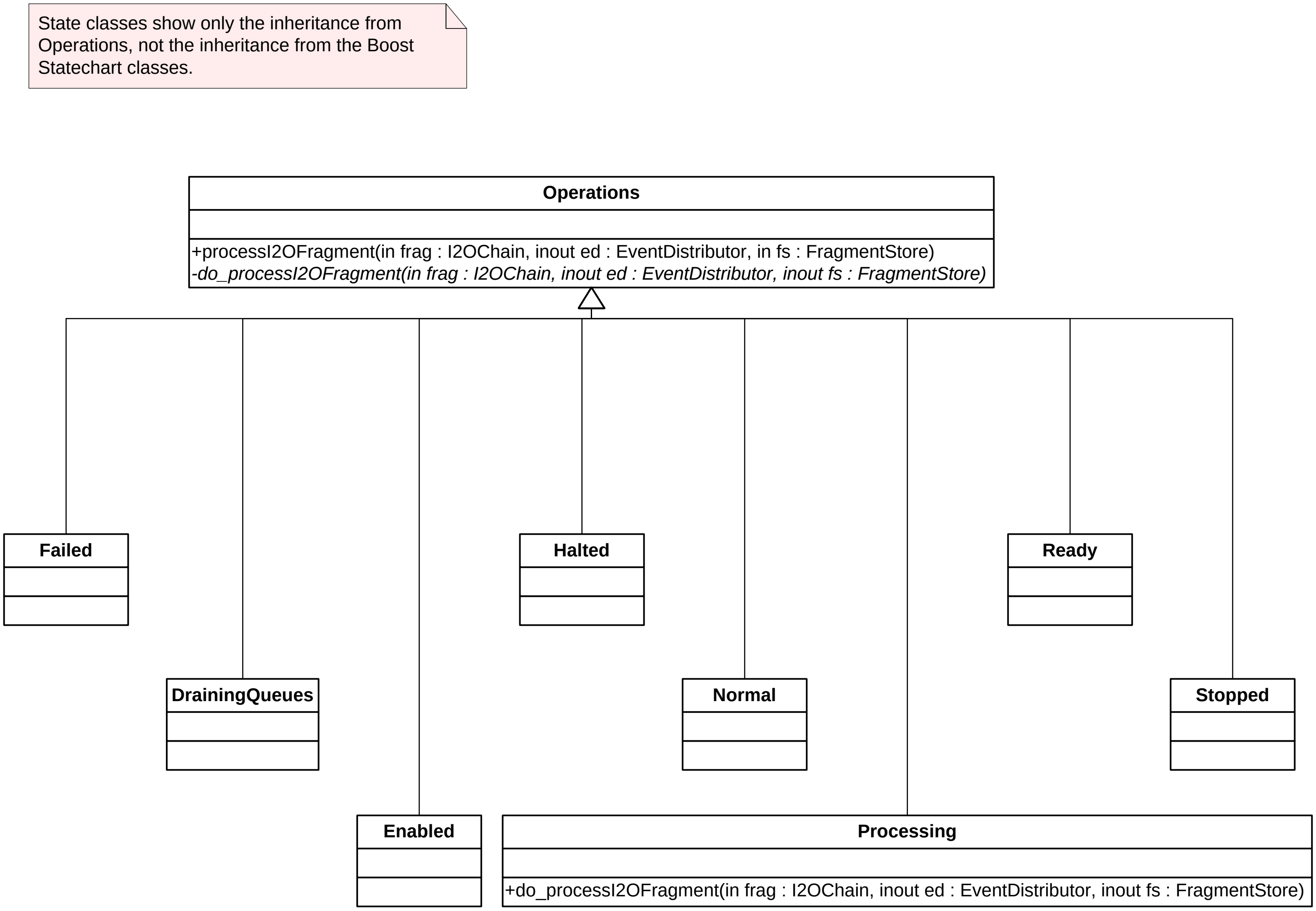
makeEventSelector uses the given EventConsumerRegistrationInfo
to create a new EventSelector object; this object is used to mark events
pass the selection criteria it encodes.

Responsible for tagging
events for routing.

Responsible for routing
events according to
their tags.

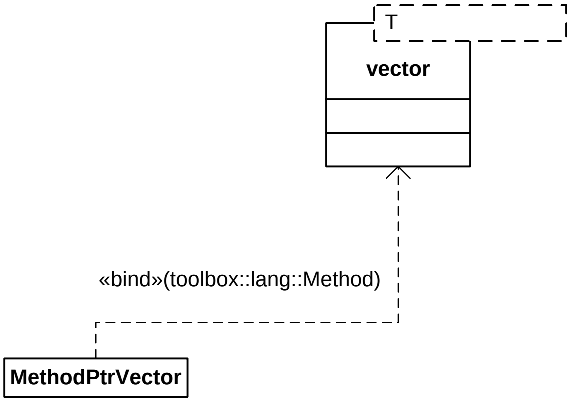
EventDistributor
+addEventToRelevantQueues(in event : I2OChain)
+full() : bool
+registerEventConsumer(in ri : EventConsumerRegistrationInfo) : QueueID
-makeEventSelector(in queue_id : QueueID, in ri : EventConsumerRegistrationInfo)
+registerEventStreams()
EventConsumerQueueCollection
+addEvent(in event : I2OChain)
+popEvent(in id) : I2OChain
+registerEventConsumer(in ri : EventConsumerRegistrationInfo) : QueueID
EventSelector
+markEvent(inout event : I2OChain)
1
*
EventConsumerQueue
+addEvent(in event : I2OChain)
+popEvent() : I2OChain
StreamQueue
+addEvent(in event : I2OChain)
+popEvent() : I2OChain
DQMEventQueue
+addEvent(in event : I2OChain)
+popEvent() : I2OChain
DiskWriter
+writeNextEvent()
+configureStreams(in configString : string)
+makeEventStreams()
+parseEventStreamConfig()
+destroyStreams()


DQMEventProcessor
+processNextDQMEvent()
+registerDQMEventConsumer(in ri : DQMEventConsumerRegistrationInfo) : QueueID
+makeDQMEventSelector(in queue_id : QueueID, in ri : DQMEventConsumerRegistrationInfo)
«uses»
DQMEventConsumerQueueCollection
+addEvent(in event : I2OChain)
+disposeOfStaleStuff()
+registerDQMEventConsumer(in ri : DQMEventConsumerRegistrationInfo) : QueueID
1
DQMEventQueue
+addEvent(in event : I2OChain)
+popEvent() : I2OChain
DQMEventConsumerRegistrationInfo
*
DQMEventConsumerQueue
+addEvent(in event : I2OChain)



State classes show only the inheritance from Operations, not the inheritance from the Boost Statechart classes.

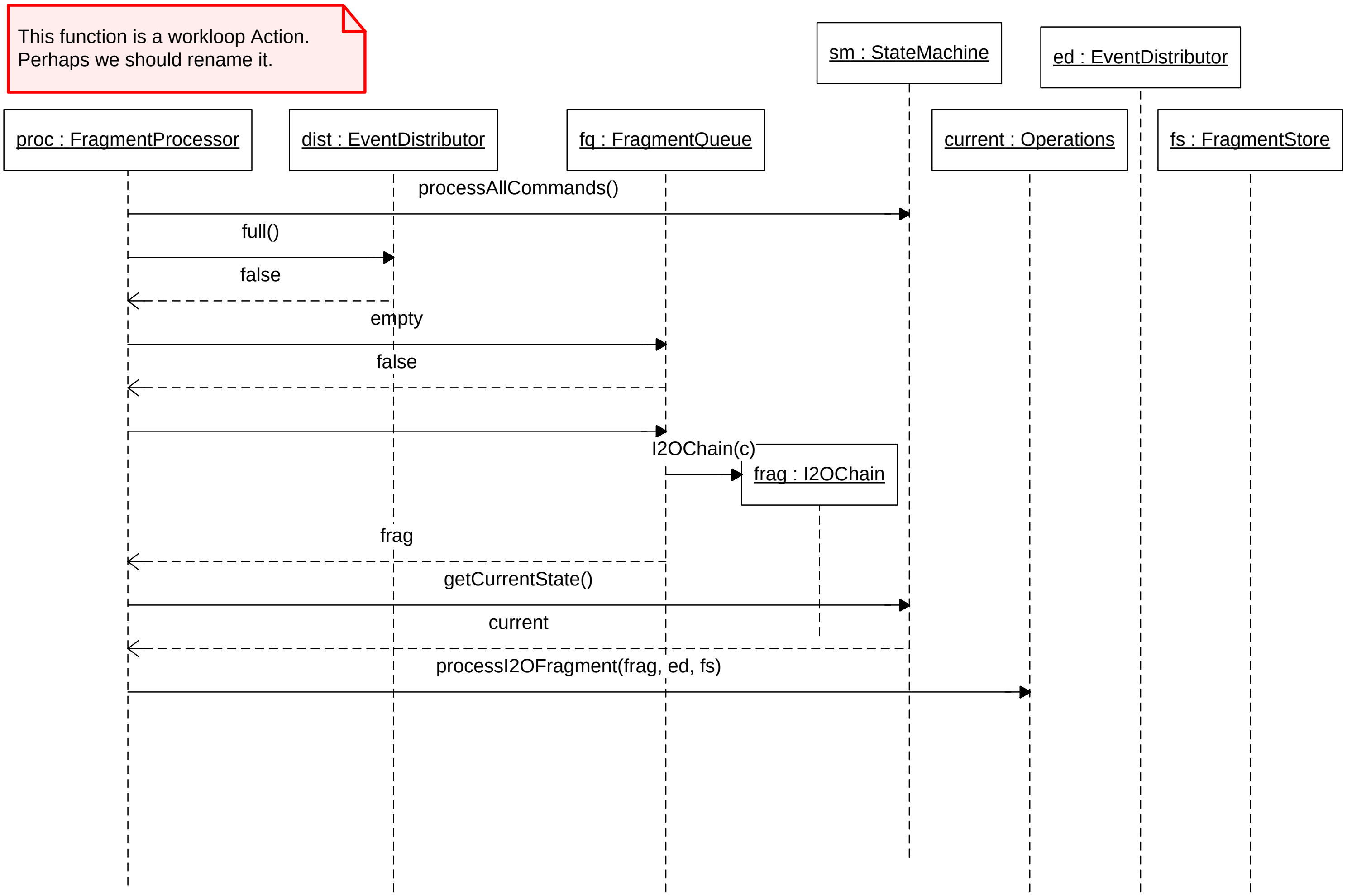
Operations
+processI2OFragment(in frag : I2OChain, inout ed : EventDistributor, in fs : FragmentStore)
-do_processI2OFragment(in frag : I2OChain, inout ed : EventDistributor, inout fs : FragmentStore)
Failed
Halted
Ready
DrainingQueues
Normal
Stopped
Enabled
Processing
+do_processI2OFragment(in frag : I2OChain, inout ed : EventDistributor, inout fs : FragmentStore)


T
vector
«bind»(toolbox::lang::Method)
MethodPtrVector

Input

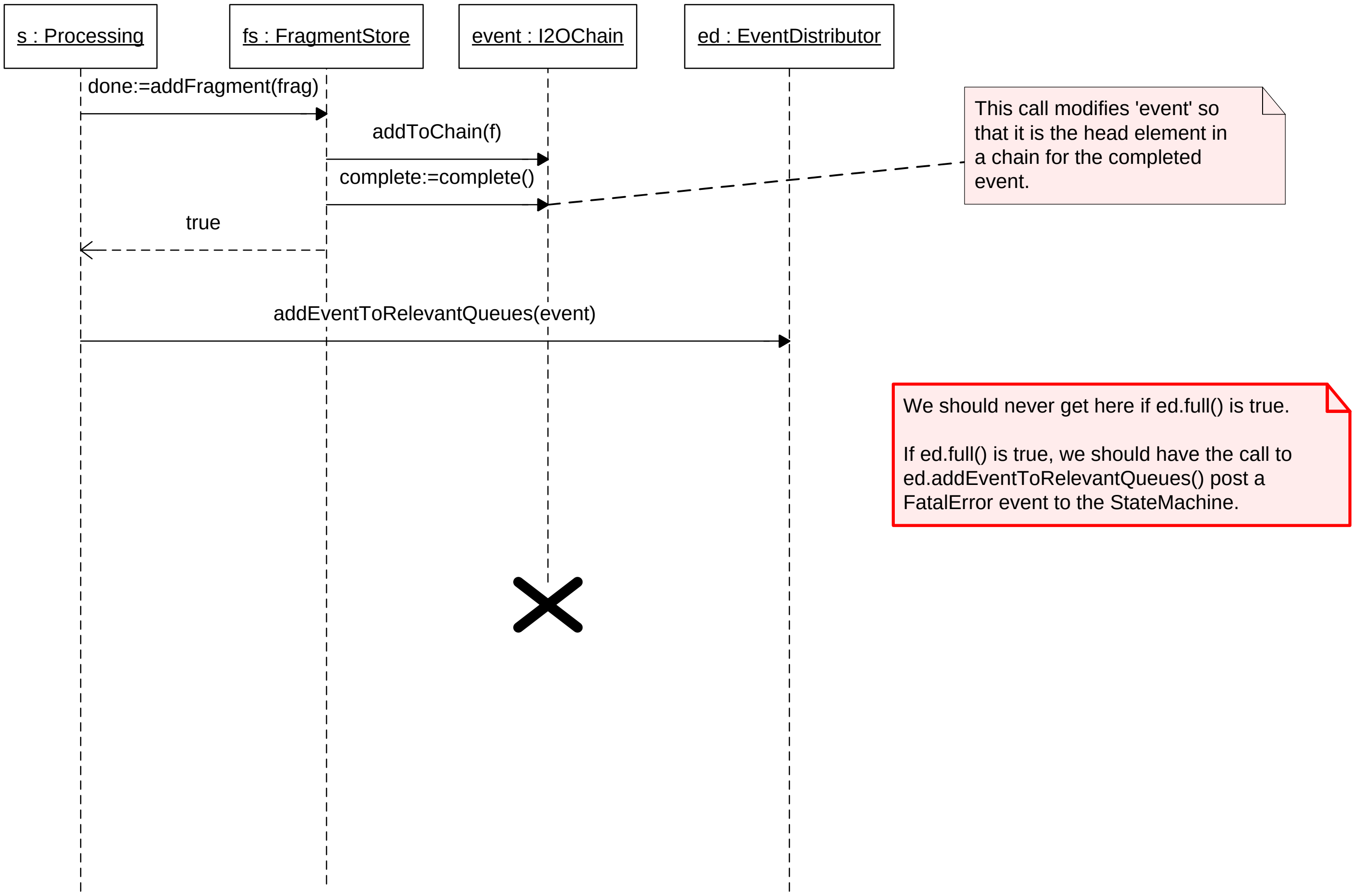
Output

FragmentProcessor::processMessages

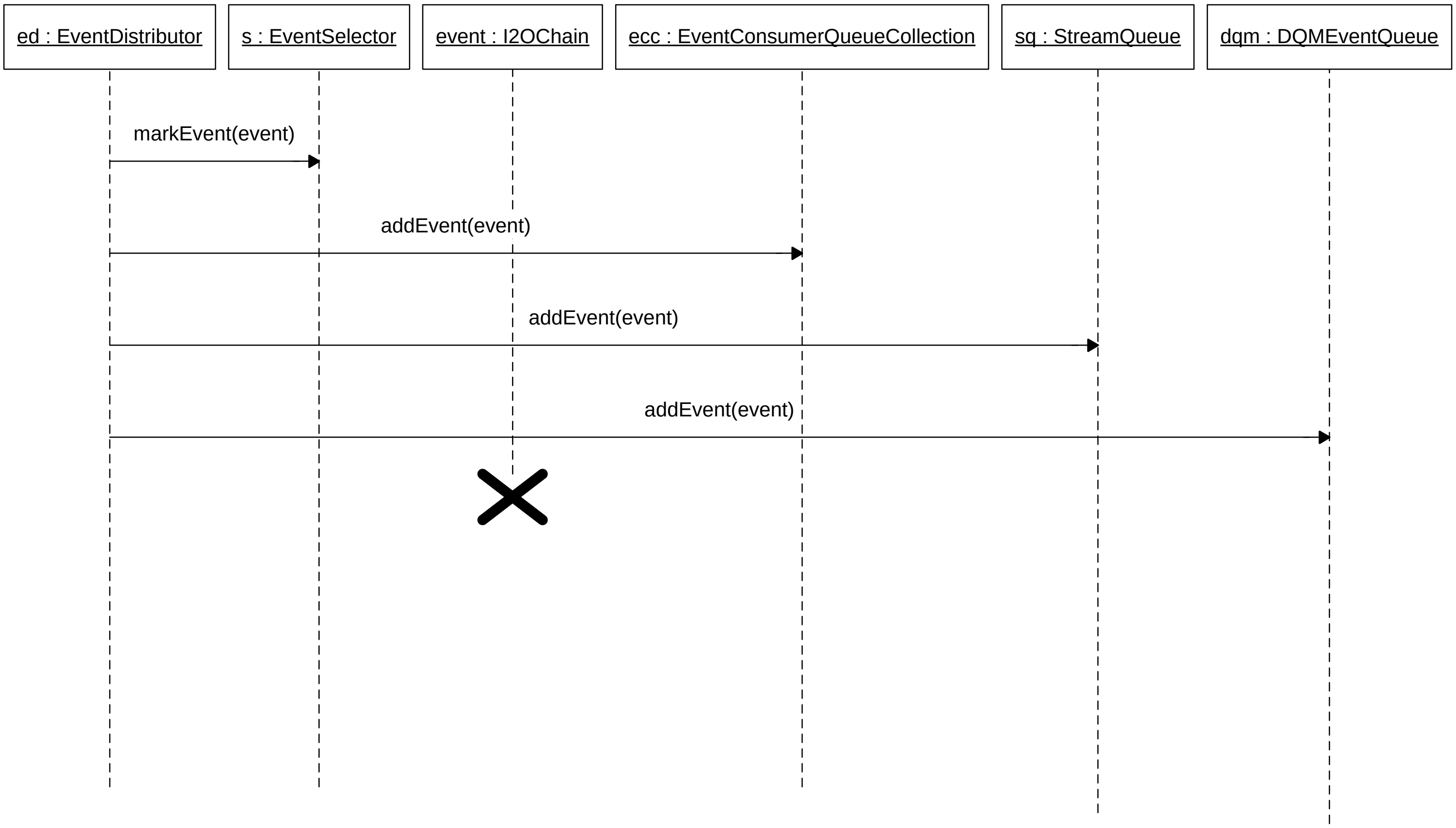


This function is a workloop Action.
Perhaps we should rename it.

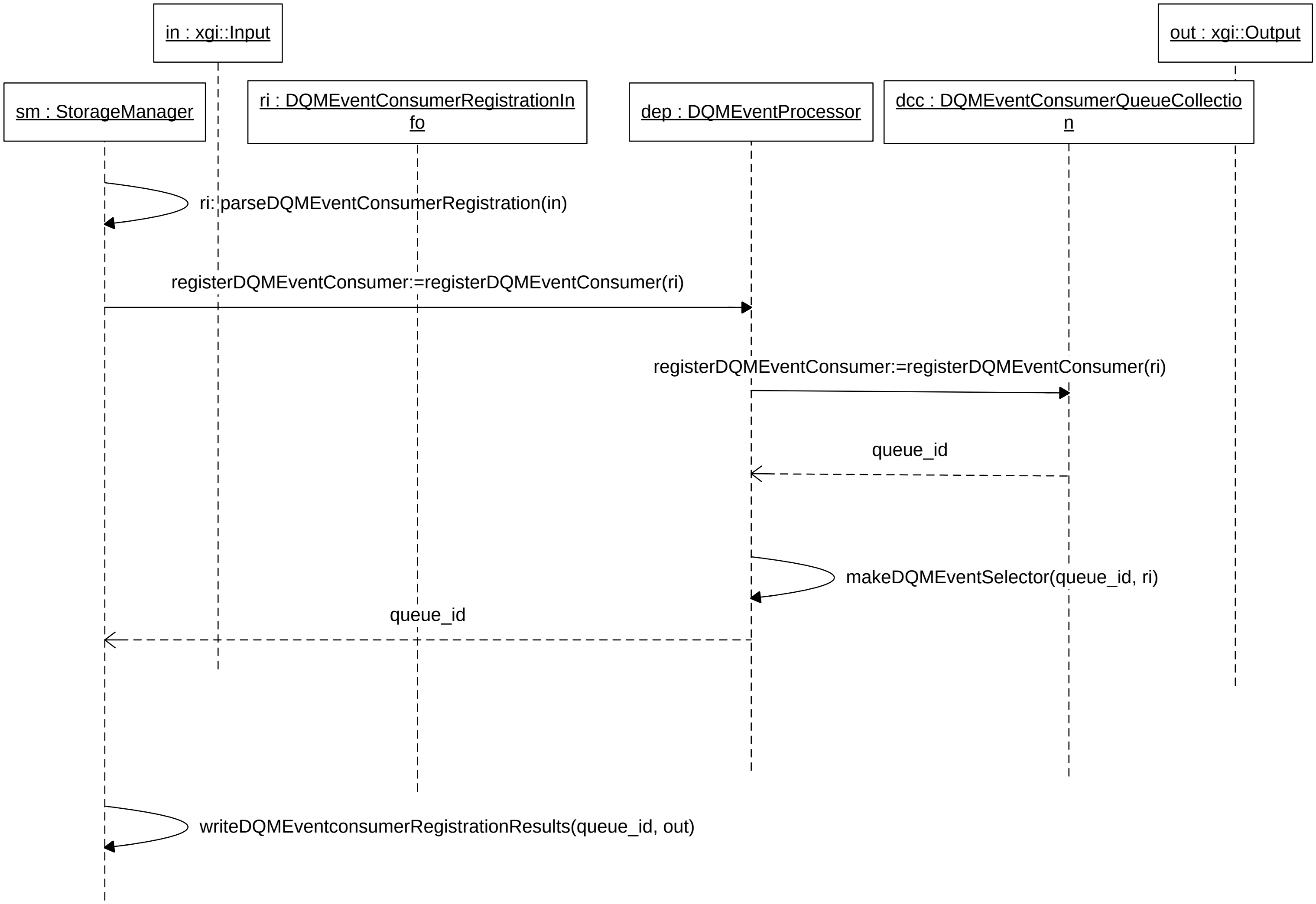
Processing State processing a fragment: last fragment case (do_processI2OFragment)


s : Processing
fs : FragmentStore
event : I2OChain
ed : EventDistributor
done:=addFragment(frag)
addToChain(f)
complete:=complete()
true
addEventToRelevantQueues(event)
This call modifies 'event' so that it is the head element in a chain for the completed event.
We should never get here if ed.full() is true.
If ed.full() is true, we should have the call to ed.addEventToRelevantQueues() post a FatalError event to the StateMachine.

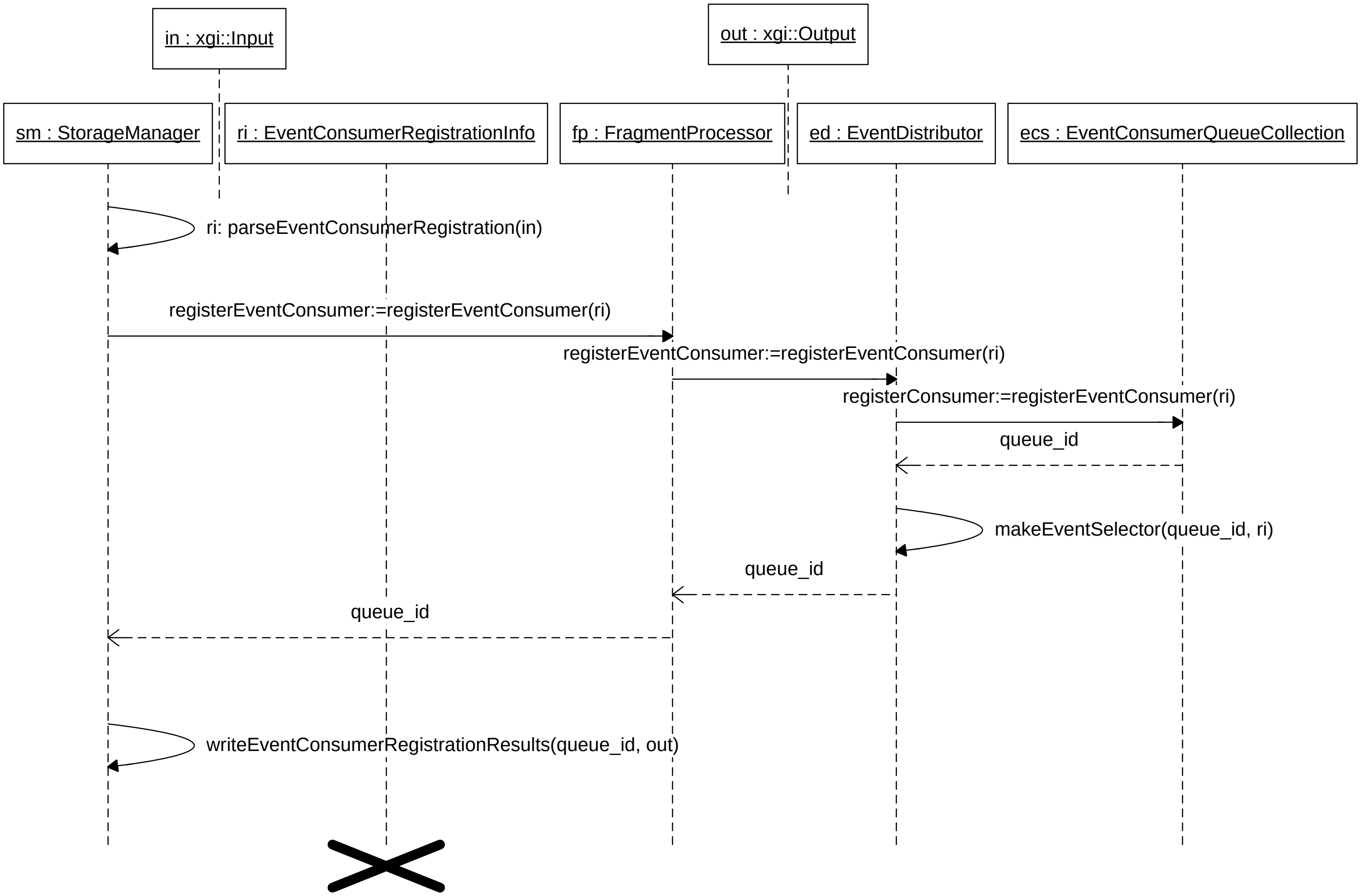
EventDistributor::addEventToRelevantQueues



DQMEvent consumer registration (StorageManager::registerDQMEventConsumer)

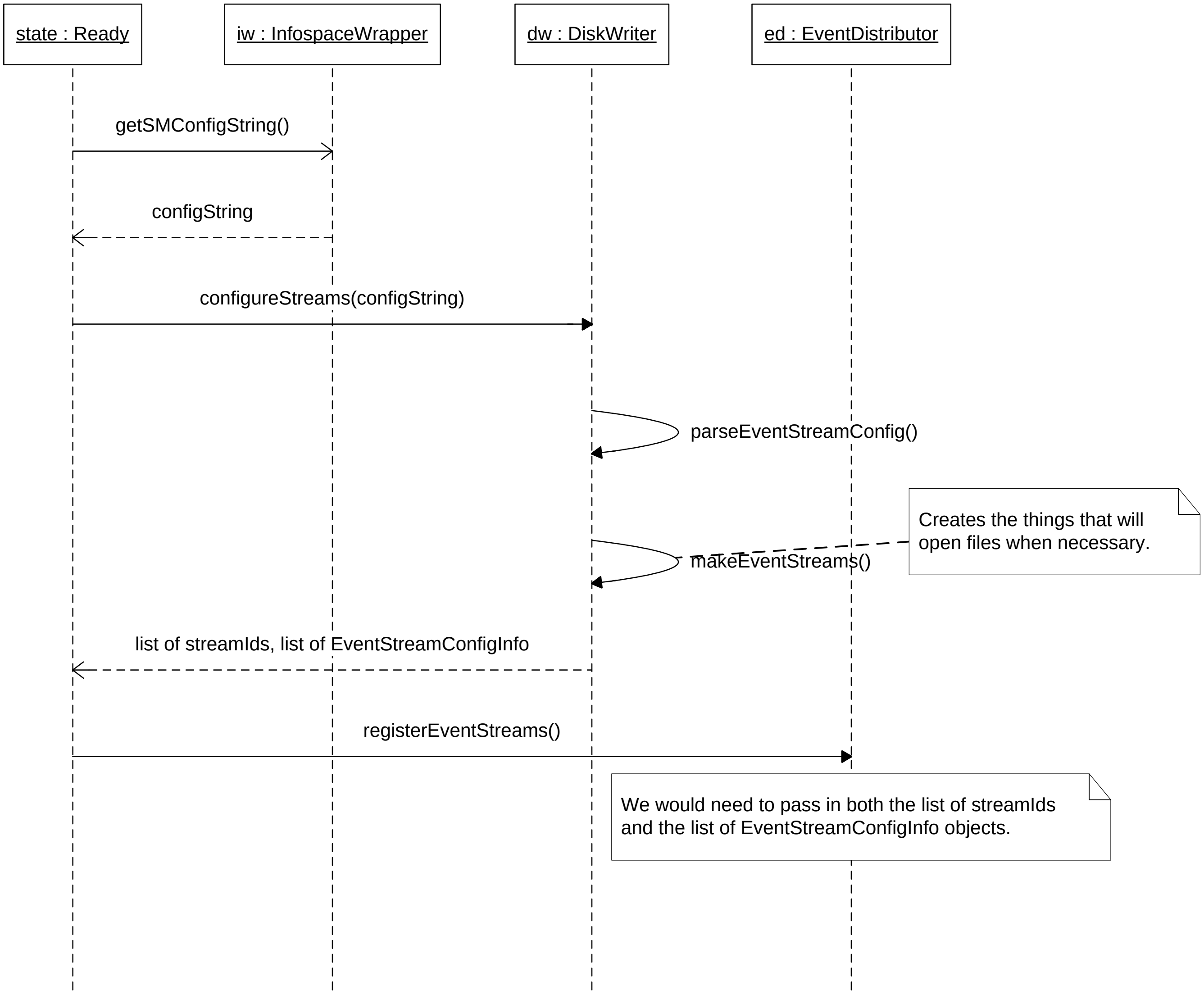

in : xgi::Input
out : xgi::Output
sm : StorageManager
ri : DQMEventConsumerRegistrationInfo
dep : DQMEventProcessor
dcc : DQMEventConsumerQueueCollection
ri: parseDQMEventConsumerRegistration(in)
registerDQMEventConsumer:=registerDQMEventConsumer(ri)
registerDQMEventConsumer:=registerDQMEventConsumer(ri)
queue_id
makeDQMEventSelector(queue_id, ri)
queue_id
writeDQMEventconsumerRegistrationResults(queue_id, out)

Event consumer registration (StorageManager::registerEventConsumer)


in : xgi::Input
out : xgi::Output
sm : StorageManager
ri : EventConsumerRegistrationInfo
fp : FragmentProcessor
ed : EventDistributor
ecs : EventConsumerQueueCollection
ri: parseEventConsumerRegistration(in)
registerEventConsumer:=registerEventConsumer(ri)
registerEventConsumer:=registerEventConsumer(ri)
registerConsumer:=registerEventConsumer(ri)
queue_id
makeEventSelector(queue_id, ri)
queue_id
queue_id
writeEventConsumerRegistrationResults(queue_id, out)

We still need to determine
how timeouts for deregistration
are handled.

Event stream configuration
(Ready entry action)



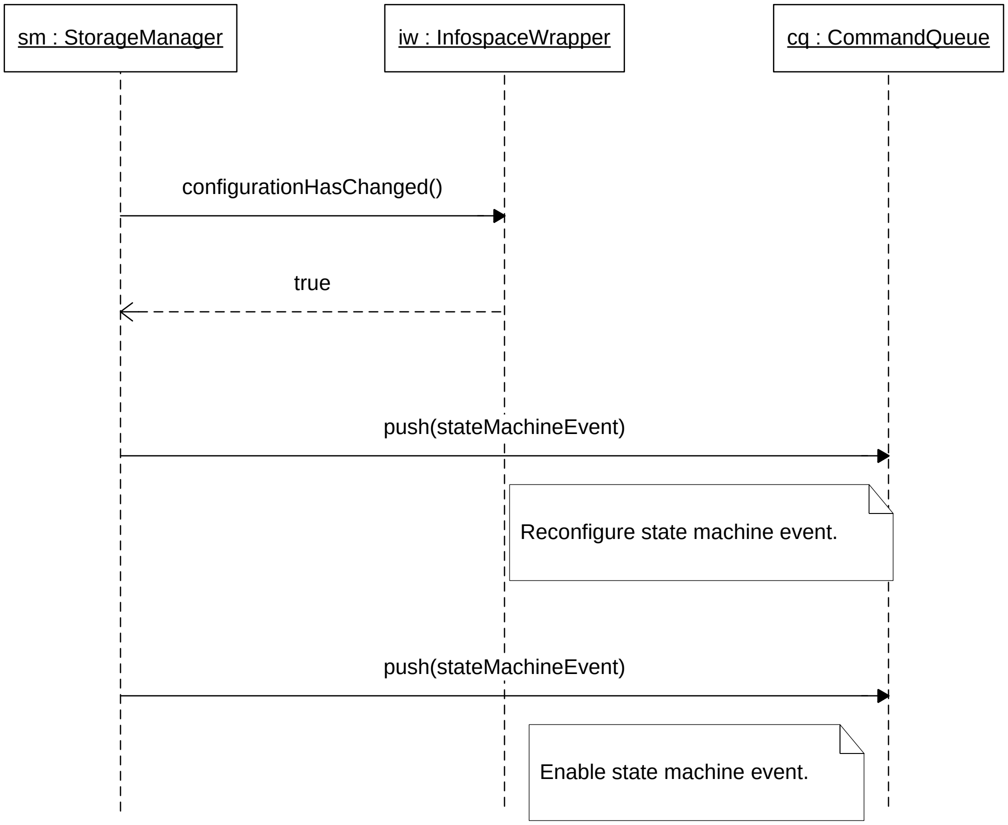
Creates the things that will open files when necessary.

We would need to pass in both the list of streamIds and the list of EventStreamConfigInfo objects.

Should we move the parsing of the config string into the EventDistrbitor, move that operation to an early part of the sequence, and pass the list of EventStreamConfigInfo objects to the DiskWriter?

We need to figure out how to make the Ready state aware of the other objects that it needs (the InfospaceWrapper, the DiskWriter, and possibly the EventDistributor).

SM callback for Enable SOAP command


sm : StorageManager
iw : InfospaceWrapper
cq : CommandQueue
configurationHasChanged()
true
push(stateMachineEvent)
Reconfigure state machine event.
push(stateMachineEvent)
Enable state machine event.