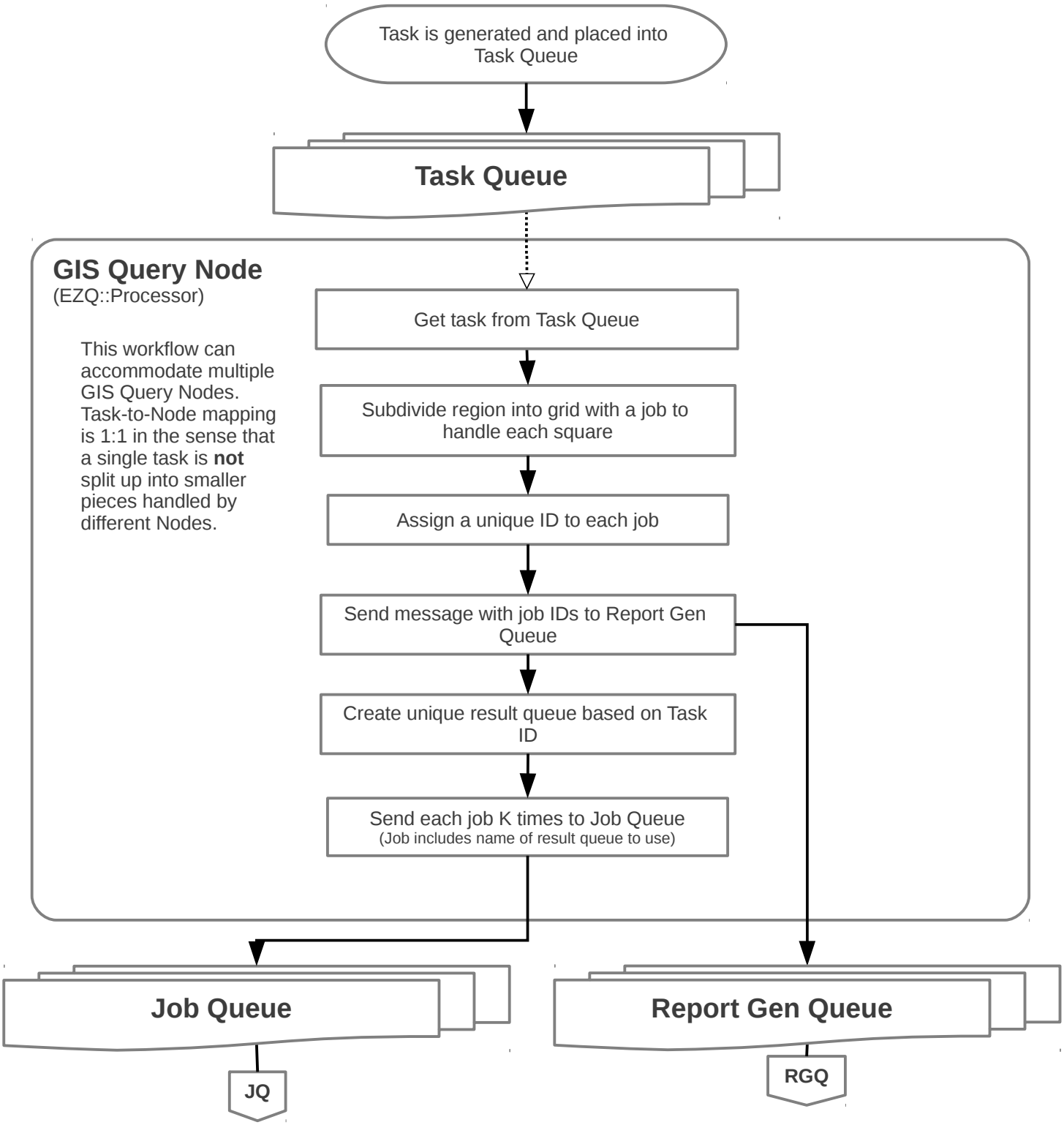
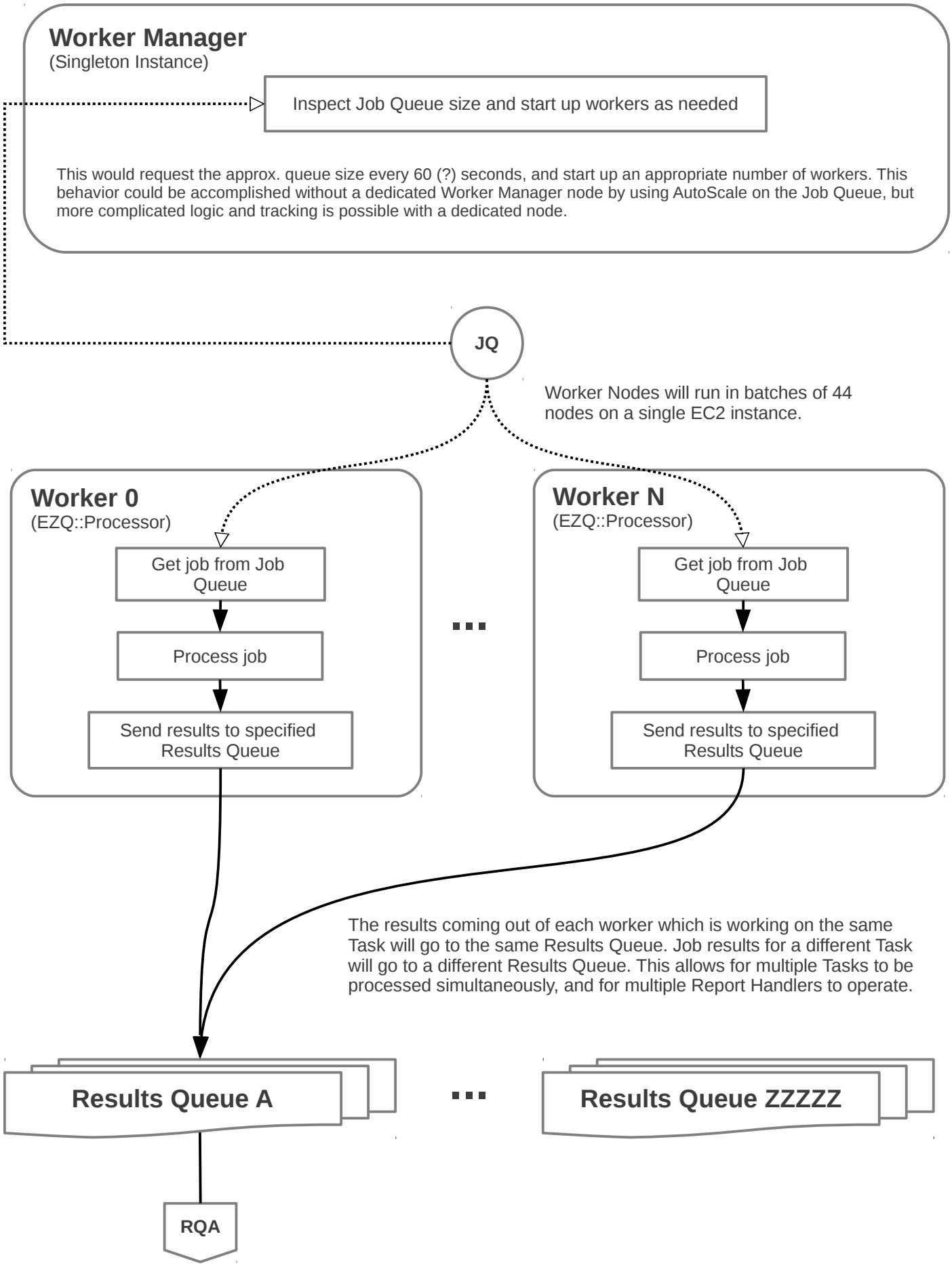
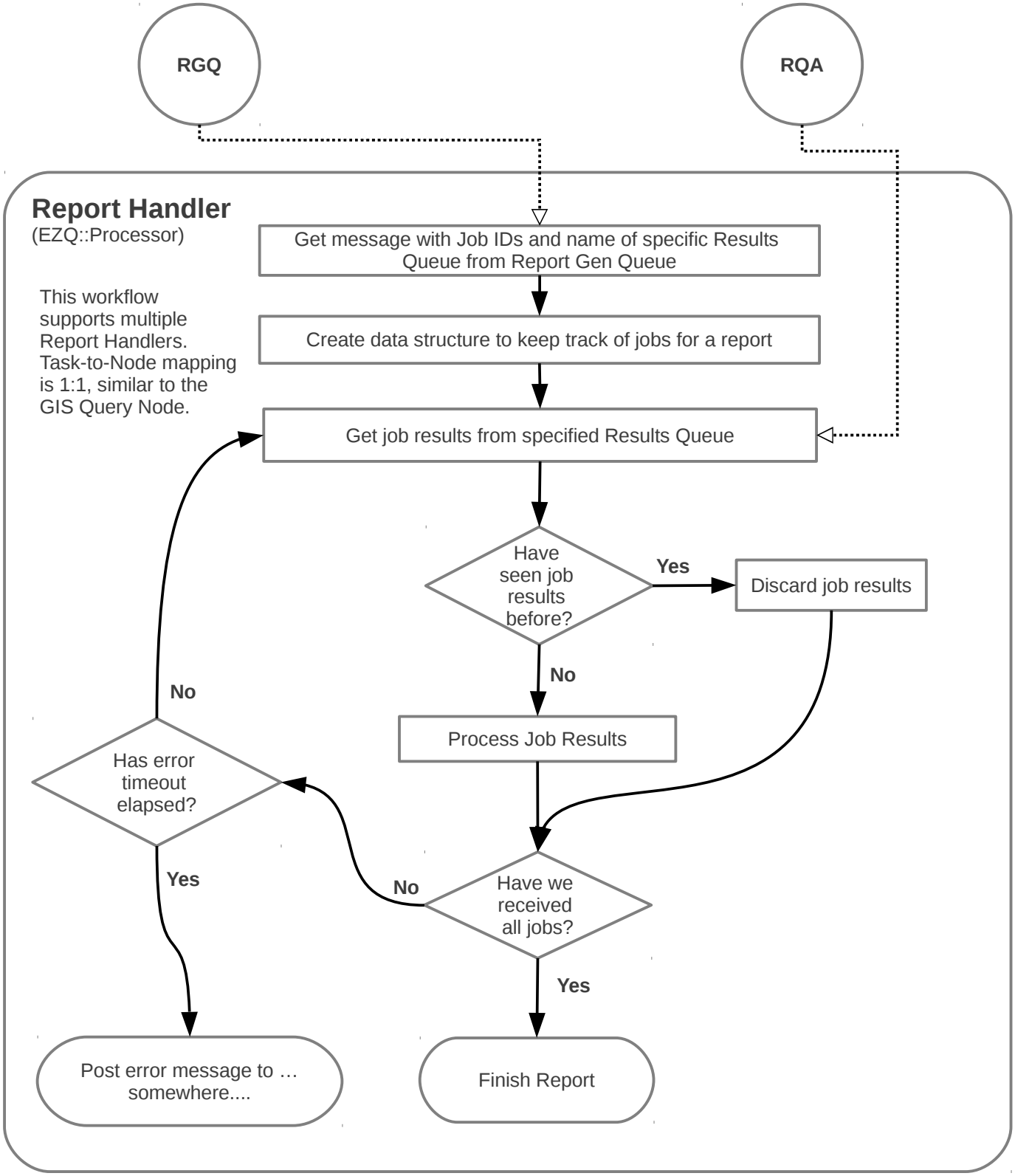
# Modified workflow #1 for 6k worker case

This workflow replaces direct network connections with queues, and uses processes wrapped inside an EZQ::Processor to handle queued messages. The queuing provides task/job/etc. persistence in the case of node failures, as well as allowing easy scaling via running multiple instances of GIS Query Node and Report Handlers.

**A**⋯▷ **B** in the following diagram indicates that B has requested and received data from A.




Worker Manager
(Singleton Instance)
Inspect Job Queue size and start up workers as needed
This would request the approx. queue size every 60 (?) seconds, and start up an appropriate number of workers. This behavior could be accomplished without a dedicated Worker Manager node by using AutoScale on the Job Queue, but more complicated logic and tracking is possible with a dedicated node.
JQ
Worker Nodes will run in batches of 44 nodes on a single EC2 instance.
Worker 0
(EZQ::Processor)
Get job from Job Queue
Process job
Send results to specified Results Queue
...
Worker N
(EZQ::Processor)
Get job from Job Queue
Process job
Send results to specified Results Queue
The results coming out of each worker which is working on the same Task will go to the same Results Queue. Job results for a different Task will go to a different Results Queue. This allows for multiple Tasks to be processed simultaneously, and for multiple Report Handlers to operate.
Results Queue A
...
Results Queue ZZZZZ
RQA


RGQ
RQA
Report Handler
(EZQ::Processor)
This workflow supports multiple Report Handlers. Task-to-Node mapping is 1:1, similar to the GIS Query Node.
Get message with Job IDs and name of specific Results Queue from Report Gen Queue
Create data structure to keep track of jobs for a report
Get job results from specified Results Queue
Have seen job results before?
Yes
Discard job results
No
Process Job Results
Has error timeout elapsed?
No
Yes
Have we received all jobs?
No
Yes
Post error message to … somewhere....
Finish Report