RabbitMQ RPC Timeout Runbook Pack

Diagnosing oslo.messaging MessagingTimeout and missed heartbeats across OpenStack. DevOps AI ToolKit · devopsaitoolkit.com

OpenStack services talk to each other over RPC on RabbitMQ via oslo.messaging: an API/worker publishes a request to a topic exchange and waits on a reply queue. A `MessagingTimeout` means the reply never came in time — the consumer is dead/slow, the queue is backed up, heartbeats are missing, or the connection is blocked. Diagnose the **queue and the consumer**, not just the service that raised the error.

> Assumes a Kolla-Ansible deployment. Read-only checks first; restart with a change record.

1. OpenStack RPC Timeout Checklist

1. **Read the error.** `MessagingTimeout` names a target (e.g. `cinder-scheduler`, `q-l3-plugin`). The *caller* logged it; the *callee* is the suspect.
2. **RabbitMQ up and un-partitioned?** `cluster_status`, alarms, partitions.
3. **Is the callee consuming?** Its queue should have `consumers > 0` and `messages` near 0.
4. **Heartbeats.** Missed heartbeats → the service silently dropped its AMQP connection.
5. **Blocked connections.** Memory/disk watermark → publishers blocked → everything times out.
6. **oslo.messaging config.** `transport_url`, `rpc_response_timeout`, heartbeat settings sane and consistent.
7. **Act narrowly.** Restart the callee (drains/re-declares its queues) before touching RabbitMQ.

2. RabbitMQ Cluster Health Checks

```
docker exec rabbitmq rabbitmqctl cluster_status
docker exec rabbitmq rabbitmqctl list_connections name state | grep -i blocked
docker exec rabbitmq rabbitmqctl status | sed -n '/alarms/,/}/p'
docker logs --tail=150 rabbitmq 2>&1 | grep -Ei "partition|missed heartbeats|closing AMQP|memory|disk"
```

- Any `partitions` listed → split-brain. Do **not** blind-restart; recover deliberately (pause_minority / restart the minority side).
- `alarms` set (memory/disk) → §5 backpressure.

3. Queue Depth Commands

```
# Backed-up or consumer-less queues (the smoking gun)
docker exec rabbitmq rabbitmqctl list_queues name messages messages_ready consumers \
  | awk 'NR>1 && ($2>100 || $4==0) {print}'

# Per-service reply queues (grow when a caller is stuck waiting)
docker exec rabbitmq rabbitmqctl list_queues name messages | grep -Ei "reply_|_fanout|scheduler|conductor|l3|dhcp"

# Overall message rates
docker exec rabbitmq rabbitmqctl list_queues | awk '{s+=$2} END{print "total messages:",s}'
```

- `messages` high + `consumers = 0` → the consuming service is down or disconnected → restart it (§7).
- `messages_ready` climbing steadily → consumers too slow / too few; scale or unblock them.

4. Consumer / Publisher Checks

```
# Who is actually connected and consuming?
docker exec rabbitmq rabbitmqctl list_consumers | awk '{print $1}' | sort | uniq -c | sort -rn
docker exec rabbitmq rabbitmqctl list_connections user peer_host state | sort | uniq -c

# Confirm the suspect service holds a connection (example: cinder)
docker exec rabbitmq rabbitmqctl list_connections name user | grep -i openstack
```

- A service with **no** consumer on its queue = it never (re)subscribed → restart it.
- Many short-lived connections churning → the service is crash-looping; check its own log.

5. Missed Heartbeat Troubleshooting

```
docker logs --tail=200 rabbitmq 2>&1 | grep -i "missed heartbeats from client"
# Map the client peer_host/port from the log to the offending service container.
grep -R "heartbeat" /etc/kolla/*/*.conf 2>/dev/null | grep -i "\[oslo_messaging_rabbit\]" -A2
```

- Missed heartbeats usually mean: an overloaded service event loop (Python GIL blocked), packet loss between service and RabbitMQ, or a heartbeat interval that's too aggressive for the load.
- oslo default heartbeat is 60s (checked at 1/2 interval). Very low values on a busy node cause false drops; extremely high values delay dead-peer detection.

6. oslo.messaging Config Checks

```
# transport_url should list ALL rabbit nodes; response timeout should be sane
grep -R "transport_url" /etc/kolla/*/*.conf | head
grep -R "rpc_response_timeout" /etc/kolla/*/*.conf     # default 60; raising it hides slowness
grep -RA3 "\[oslo_messaging_rabbit\]" /etc/kolla/*/*.conf | grep -Ei "heartbeat|rabbit_ha|amqp"
```

- All services should point at the **same** RabbitMQ endpoints. A single-node `transport_url` in an HA cluster loses failover.
- Raising `rpc_response_timeout` is a stopgap, not a fix — find the slow callee.

7. Service Restart Decision Tree

> **Restart the consumer, not the messenger.** RabbitMQ is stateful and clustered; restarting a service re-declares its queues and reconnects, which fixes most "consumers = 0" cases without risking the broker.

```
MessagingTimeout to <service>?
 ├─ RabbitMQ partitioned or alarmed? ── yes → fix broker FIRST (§2/§5), escalate; do not restart services blindly
 └─ no
     ├─ <service> queue has consumers=0? ── yes → docker restart <service>  (e.g. cinder_scheduler)
     ├─ queue backed up but consumers>0? ── yes → consumer too slow: check its CPU/log, scale workers
     └─ heartbeats missing only?         ── yes → docker restart <service>; if it recurs, tune heartbeat / check network
```

```
# Targeted restarts (Kolla container names)
docker restart cinder_scheduler          # or nova_conductor / neutron_l3_agent / heat_engine
docker restart nova_conductor
# Broker is a last resort and single-node only:
docker restart rabbitmq                   # NEVER on a partitioned cluster
```

8. Nova / Cinder / Neutron / Heat Symptom Matrix

Service	Timeout looks like	Prime suspect queue / target
Nova	`openstack server create` stuck in BUILD; `nova-conductor` MessagingTimeout	`conductor`, compute `_fanout` on the target host
Cinder	Volume stuck `creating`; `cinder-scheduler` MessagingTimeout; "No valid backend"	`cinder-scheduler`, `cinder-volume.<host>`
Neutron	Ports `BUILD`, floating IPs down; l3/dhcp agent `XXX`	`q-l3-plugin`, `q-plugin`, agent report_state
Heat	Stacks stuck `IN_PROGRESS`; `heat-engine` RPC timeout	`engine`, `engine_worker`

9. Copy/Paste One-Shot Triage

```
echo "== cluster ==";      docker exec rabbitmq rabbitmqctl cluster_status | grep -Ei "running_nodes|partitions"
echo "== alarms ==";       docker exec rabbitmq rabbitmqctl status | sed -n '/alarms/,/}/p'
echo "== blocked ==";      docker exec rabbitmq rabbitmqctl list_connections state | grep -c blocked
echo "== hot queues ==";   docker exec rabbitmq rabbitmqctl list_queues name messages consumers | awk 'NR>1 && ($2>100||$3==0){print}'
echo "== heartbeats ==";   docker logs --tail=200 rabbitmq 2>&1 | grep -c "missed heartbeats"
```

10. Incident Notes Template

```
INCIDENT: OpenStack RPC / MessagingTimeout
Started (UTC):        ____    Detected by: ____    Sev: ____
Caller (logged err):  ____    Callee/target: ____
RabbitMQ:             [ ] running nodes __  [ ] partitions: ____  [ ] alarms: ____
Queue evidence:       name ____   messages ____   consumers ____
Heartbeats missed:    ____ (client peer_host: ____)
oslo config:          rpc_response_timeout ____   transport_url covers all nodes? Y/N
Action taken:         (service restarted / scaled / broker action + who + when)
Result:               [ ] recovered @ ____    [ ] escalated to ____
Root cause:           ____
Follow-ups:           (heartbeat tuning, worker scaling, HA transport_url, alerts) ____
```