Rsyslog End-to-End Debug

1) Basic sanity

Why: Ensure config is valid and the daemon is healthy before deeper checks.

```
sudo rsyslogd -N1
systemctl status rsyslog --no-pager
rsyslogd -v
```

LB in `omfwd` requires **v8.2408+** (8.2502+ recommended).

---

2) Inputs are listening

Why: The relay must accept logs on fixed ports.

```
ss -lntu | egrep ':514|:6514'
```

Expected listeners: **UDP 514**, **TCP 514**, **TCP 6514 (TLS)**; `impstats` is enabled for health.

---

3) Target reachability (per-backend)

Why: If a backend can't be reached on 6514/TCP, rsyslog will skip it; if *all* fail, the action suspends.

```
for H in <BACKEND_1> <BACKEND_2>; do
  timeout 3 bash -c "cat </dev/null >/dev/tcp/$H/6514" \
    && echo "$H:6514 OK" || echo "$H:6514 FAIL"
done
```

Replace placeholders with production hosts.

---

4) Outbound sockets are open

Why: Confirms rsyslog is actually connecting to the collectors.

```
ss -ntp dst :6514 | grep rsyslog || true
```

---

5) Action/queue health via `impstats` (no parser)

Why: Shows forwarding health, buffering, and suspension state.

```
# show recent metrics
tail -n 50 /var/log/rsyslog_stats.json
```

Look for the action `"actionName":"lp_tls_rr"` and fields: `failed`, `suspended`, `queuesize`.

- `suspended=true` ⇒ all targets down → buffering
- `queuesize` should **drain** after recovery
- spool lives under `/var/spool/rsyslog/` (disk-assisted queue)

---

6) Round-robin & failover smoke

Why: Proves LB continuity when one backend drops.

1. Stop one backend (or block its port).
2. Generate a test event:

```
logger -t ESA_SMOKE "RR test: one backend down"
```

3. Expect traffic to continue via the remaining target (no suspension). Restore and observe alternation.

---

7) Buffering behavior (all targets down)

Why: Confirms disk buffering and automatic replay.

1. Stop **both** backends (or block 6514 egress).
2. Send traffic:

```
for i in {1..500}; do logger -t BUFFER_TEST "msg $i"; done
```

3. Watch spool grow and later drain:

```
du -sh /var/spool/rsyslog/
ls -lh /var/spool/rsyslog/ | head
```

After restoring targets, backlog drains automatically.

---

8) Local fallback file (evidence during suspension)

Why: Keeps proof of received logs while egress is suspended.

```
tail -f /var/log/esa_fallback-buffer.log
```

This action writes **only** while the TLS action is suspended.

---

9) Firewall quick check (relay host)

Why: Missing ports = no flow.

```
sudo firewall-cmd --list-ports
# expect: 514/udp 514/tcp 6514/tcp
```

Ports policy for the relay and guidance on opening them.

---

10) End-to-end smoke

Why: Final confirmation of the full path.

```
logger -t ESA_SMOKE "rsyslog LB end-to-end OK"
```

Use after enabling the service to validate flow.

---

11) Track per-backend failures **and** suspension (logs only)

Why: Identify which backend dropped and when the action suspended/resumed — with just `/var/log/messages`.

```
# Per-backend connection changes (down/up)
grep -i 'omfwd.*connection' /var/log/messages

# Action suspended/resumed (all targets down → suspended; recovery → resumed)
grep -i 'omfwd.*suspend' /var/log/messages
grep -i 'omfwd.*resume'  /var/log/messages

# Live watch
tail -f /var/log/messages | grep -iE 'omfwd.*(connection|suspend|resume)'
```

Interpreting alongside design: single-backend failures are skipped; **all** down ⇒ action **suspended**; recovery resumes and replays backlog.

Reference (what the config does)

- Inputs + `impstats` + TLS defaults, spool location.
- TLS egress with round-robin target pool, buffering, infinite retry.
- Behavior: skip failed backend; suspend if all down; local fallback only during suspension.