Detailed Cheat Sheet for Debugging Logs in the Rsyslog Test Plan

General Debugging Tools (Applicable Everywhere):

- **Impstats**: Impstats logs are generated every 60 seconds by the `impstats` module in CEE/JSON format (Doc page 4, 8). Use `tail -f /var/log/rsyslog_stats.json` for live monitoring or `jq -r ... /var/log/rsyslog_stats.json` for static analysis. If misrouted, check `grep impstats /var/log/messages`. The `jq` tool is essential for parsing and filtering this JSON, especially under high load or during outages.
- **Buffers/Queues**: `ls -lh /var/spool/rsyslog/` (check for q_logpoint_rr* files, growth > highwatermark=40000); `du -sh /var/spool/rsyslog` (total size); Fallback: `tail -f /var/log/esa_fallback-buffer.log`.
- **Journal**: `journalctl -u rsyslog -f` (live); `journalctl -u rsyslog -e` (latest entries); `journalctl -u rsyslog --since "5 minutes ago"` (specific period).
- **Others**: `rsyslogd -d -n` (stop service first; debug in foreground); `tcpdump -i any port 6514` (egress traffic).

Explanation of jq Usage:

- **Why jq?**: Impstats produces complex JSON logs (e.g., `{"name":"action","actionName":"lp_tls_rr","submitted":10,"failed":0,"suspended":false,"queuesize":0}`). `jq` is a command-line JSON processor that extracts specific fields (e.g., `submitted`, `queuesize`) for quick analysis, especially under high load or multi-step scenarios (outage, recovery). Without `jq`, manual parsing becomes impractical with large datasets.
- **Basic Command**:
  - Static Analysis: `jq -r 'select(.name=="action" and .actionName=="lp_tls_rr") | "\(.timegenerated) submitted=\(.submitted) failed=\(.failed) suspended=\(.suspended) queuesize=\(.queuesize)"' /var/log/rsyslog_stats.json`
  - Live Monitoring: `tail -f /var/log/rsyslog_stats.json | jq -r 'select(.name=="action" and .actionName=="lp_tls_rr") | "\(.timegenerated) submitted=\(.submitted) failed=\(.failed) suspended=\(.suspended) queuesize=\(.queuesize)"'`
- **Specific Examples**:
  - Count processed messages: `jq -r 'select(.submitted>0) | .submitted' /var/log/rsyslog_stats.json` (static) or `tail -f /var/log/rsyslog_stats.json | jq -r 'select(.submitted>0) | .submitted'` (live).
  - Maximum queue size: `jq -r 'max_by(.queuesize) | .queuesize' /var/log/rsyslog_stats.json`.
  - Suspended state: `jq -r 'select(.suspended==true) | .queuesize' /var/log/rsyslog_stats.json`.
- **Installation**: If missing, `dnf install jq -y`.
- **Use Cases**: Varies by phase. Static analysis (post-test) uses file input; live monitoring (during tests) uses `tail -f`. Adjust filters (e.g., `.timegenerated`, `.failed`) based on scenario.

General Tips:

- Check all three logs (impstats, buffers, journal) after each test for consistency.
- If impstats misrouted: Verify `if $syslogtag == 'impstats:' then omfile + stop` rule (Doc page 5).
- If buffers empty: Check highwatermark not reached (generate more traffic) or /var/spool permissions (700 root).
- SELinux/Firewall: `ausearch -m avc -ts recent`; `firewall-cmd --list-all`.
- Restart: `systemctl restart rsyslog`; Wait 60s for impstats.

Debugging by Test Plan Phase

Phase 1: Prerequisites Verification

Situations: OS, FIPS, network, TLS checks (pre-traffic setup).

- **What to Verify**: No active logs yet; focus on journal for initial errors (e.g., FIPS/TLS impact).
- **Commands**:
  - Impstats: N/A (not loaded).
  - Buffers: N/A.
  - Journal: `journalctl -u rsyslog -e` (initial errors if service running).
  - jq: N/A.
- **What Can Go Wrong**: FIPS blocks GnuTLS; debug: `rsyslogd -d -n` for crypto (Doc page 9). Solution: Temporarily disable FIPS.

Phase 2: Installation Validation

Situations: Install repo/RPM, version check (installation errors logged).

- **What to Verify**: Journal for missing dependencies.
- **Commands**:
  - Impstats: N/A.
  - Buffers: N/A.
  - Journal: `journalctl -u dnf`; Post-install: `journalctl -u rsyslog`.
  - jq: N/A.
- **What Can Go Wrong**: GPG signature fail; debug: `rpm -K *.rpm` (Doc page 3). Solution: Import key.

Phase 3: Configuration Setup and Syntax Check

Situations: Config edits, syntax check (parsing errors logged).

- **What to Verify**: Journal for parse errors; impstats rule readiness.
- **Commands**:
  - Impstats: Post-restart: `tail /var/log/rsyslog_stats.json`; jq: N/A (no data yet).
  - Buffers: `ls /var/spool/rsyslog` (dir ready).
  - Journal: `journalctl -u rsyslog -e`.
  - jq: N/A.
- **What Can Go Wrong**: Bad regex in filters (Doc page 3); debug: `rsyslogd -N1` + `rsyslogd -d -n | grep re_match`. Solution: Fix regex.

Phase 4: Customization (Backends and Filters)

Situations: Sed backends, filter setup/test (drops affect logs).

- **What to Verify**: Journal for filter errors; impstats for processed logs.
- **Commands**:
  - Impstats: `tail /var/log/rsyslog_stats.json`; jq: `jq -r 'select(.submitted>0) | .submitted' /var/log/rsyslog_stats.json` (if processed).
  - Buffers: N/A.
  - Journal: `journalctl -f` during `logger` test.
  - jq: Filter drops: `jq -r 'select(.failed>0) | .failed' /var/log/rsyslog_stats.json` (if filter misconfig).
- **What Can Go Wrong**: All messages dropped (bad regex); debug: Temp omfile (Doc page 3). Solution: Adjust blacklist.d.

Phase 5: Service Activation and Basic Functionality

Situations: Service start, smoke test (basic traffic).

- **What to Verify**: Impstats submitted>0; journal OK; buffers empty (normal).
- **Commands**:
  - Impstats: `tail -f /var/log/rsyslog_stats.json | jq -r 'select(.submitted>0) | .submitted'` (live count); or `jq -r 'select(.submitted>0) | .submitted' /var/log/rsyslog_stats.json` (static).
  - Buffers: `ls /var/spool/rsyslog`.
  - Journal: `journalctl -u rsyslog -f`.
  - jq: Activity: `jq -r '.[].timegenerated' /var/log/rsyslog_stats.json`.
- **What Can Go Wrong**: No forwarding (TLS fail); debug: `rsyslogd -d -n` (TLS handshake). Solution: Check certs (Doc page 2).

Phase 6: Input and Forwarding Tests

Situations: Test TCP 514 (normal forwarding).

- **What to Verify**: Impstats submitted rising; journal message flow.
- **Commands**:
  - Impstats: `jq -r 'select(.submitted>0) | .submitted' /var/log/rsyslog_stats.json` (per test); or `tail -f /var/log/rsyslog_stats.json | jq -r 'select(.submitted>0) | .submitted'` (live).
  - Buffers: Empty.
  - Journal: `journalctl -u rsyslog -e | grep "TCP test"`.
  - jq: Trend: `jq -r 'sort_by(.timegenerated) | .[].submitted' /var/log/rsyslog_stats.json`.
- **What Can Go Wrong**: No input; debug: `tcpdump port 514`. Solution: Check imtcp (Doc page 4).

Phase 7: Load-Balancing and Failover Tests

Situations: Both up (RR), one down (failover), resume, high-load failover (partial outage, low queues).

- **What to Verify**: Impstats failed briefly (one down), submitted rising; journal retries; buffers minimal.
- **Commands**:

    - Impstats: `tail -f /var/log/rsyslog_stats.json | jq -r 'select(.failed>0) | .failed'` (one down); `jq -r 'select(.suspended==false) | .submitted' /var/log/rsyslog_stats.json` (RR).
    - Buffers: `ls /var/spool` (minimal if high-load).
    - Journal: `journalctl -f | grep resume`.
    - jq: High-load: `jq -r 'max_by(.queuesize) | .queuesize' /var/log/rsyslog_stats.json` (low).
- **What Can Go Wrong**: No failover (bad pool); debug: `tcpdump` + `rsyslogd -d -n`. Solution: Verify config (Doc page 6).

Phase 8: Buffering and Queue Tests

Situations: Both down (buffer activation), draining, fallback, max limit, filters+buffer.

- **What to Verify**: Impstats suspended=true, queuesize>40000; buffers grow/drain; journal suspend/resume.
- **Commands**:
    - Impstats: `jq -r 'select(.suspended==true) | .queuesize' /var/log/rsyslog_stats.json` (down); `tail -f /var/log/rsyslog_stats.json | jq -r 'select(.suspended==false and .queuesize==0)'` (drain).
    - Buffers: `watch -n 1 'ls -lh /var/spool/rsyslog'`; `tail -f /var/log/esa_fallback-buffer.log`.
    - Journal: `journalctl -f | grep suspended`.
    - jq: Max: `jq -r 'max_by(.queuesize) | .queuesize' /var/log/rsyslog_stats.json` (≤50000).
- **What Can Go Wrong**: No growth; debug: Increase loop. Solution: Adjust highwatermark (Doc page 5).

Phase 9: Monitoring with Impstats

Situations: File check, summarize, live (focus).

- **What to Verify**: Impstats populated; not in messages.
- **Commands**:
    - Impstats: `tail -f /var/log/rsyslog_stats.json | jq -r '.'` (live full view); `jq -r 'select(.name=="action" and .actionName=="lp_tls_rr") | "\(.timegenerated) submitted=\(.submitted) failed=\(.failed) suspended=\(.suspended) queuesize=\(.queuesize)"' /var/log/rsyslog_stats.json` (Doc page 8).
    - Buffers: N/A.
    - Journal: `journalctl -e | grep impstats`.
    - jq: Analysis: `jq -r 'sort_by(.timegenerated) | .[].submitted' /var/log/rsyslog_stats.json`.
- **What Can Go Wrong**: Empty; debug: `rsyslogd -d -n`. Solution: Check module load (Doc page 4).

Phase 10: FIPS and TLS-Specific Tests

Situations: CLEAR failover/buffer, TLS failover.

- **What to Verify**: Journal TLS errors; impstats suspended if fail; buffers as phase 8.
- **Commands**:

    - Impstats: `jq -r 'select(.failed>0) | .failed' /var/log/rsyslog_stats.json` (TLS fail); `tail -f /var/log/rsyslog_stats.json | jq -r 'select(.failed>0) | .failed'` (live).
    - Buffers: Check during CLEAR outage.
    - Journal: `journalctl -f | grep gtls`.
    - jq: Trend: `jq -r '.[].suspended' /var/log/rsyslog_stats.json`.
- **What Can Go Wrong**: FIPS block; debug: Disable temp. Solution: Inherit policies (Doc page 9).

Phase 11: Edge Cases and Debugging

Situations: High load, buffer debug, impstats debug, retry.

- **What to Verify**: Impstats queuesize high; buffers max 10g; journal retries.
- **Commands**:
    - Impstats: `jq -r 'max_by(.queuesize) | .queuesize' /var/log/rsyslog_stats.json` (load); `tail -f /var/log/rsyslog_stats.json | jq -r 'max_by(.queuesize) | .queuesize'` (live).
    - Buffers: `du -sh /var/spool`.
    - Journal: `journalctl -f | grep retry`.
    - jq: Retry: `jq -r '.[].failed' /var/log/rsyslog_stats.json`.
- **What Can Go Wrong**: Overflow; debug: `rsyslogd -d -n`. Solution: Tune params (Doc page 5).

Phase 12: Rollback and Final Validation

Situations: Rollback, checklist, final test.

- **What to Verify**: Journal downgrade errors; impstats post-revert; buffers cleared.
- **Commands**:
    - Impstats: `jq -r '.[].submitted' /var/log/rsyslog_stats.json` (stable); `tail -f /var/log/rsyslog_stats.json | jq -r '.[].submitted'` (live).
    - Buffers: `rm -rf /var/spool/rsyslog/*`.
    - Journal: `journalctl -u rsyslog -e`.
    - jq: Verify: `jq -r 'select(.failed==0)' /var/log/rsyslog_stats.json`.
- **What Can Go Wrong**: Config residue; debug: Restore backup. Solution: Doc page 8 rollback.

Appendices:

- **SELinux**: `setenforce 0`; `ausearch -m avc`.
- **Load**: `top -p $(pgrep rsyslogd)`.
- **Empty Logs**: Check service/config.

Note: `jq` enables dynamic, precise analysis (e.g., `timegenerated` for timelines). Use static (`/var/log/rsyslog_stats.json`) for post-test reviews and `tail -f` for real-time monitoring. Adapt filters as needed (e.g., `.failed` for errors).