HEALTH CHECK HANDBOOK

VIRTUAL CONTROL

VMWARE CLOUD FOUNDATION SOLUTIONS

VCF Operations for Logs Health Check Handbook

Health validation covering cluster status, ingestion rates, storage capacity, log forwarders, content packs, and alerting configuration.

LOG INSIGHT | INGESTION | FORWARDERS | CONTENT PACKS | CASSANDRA

VCF 9.0 VMware Cloud Foundation

PROPRIETARY & CONFIDENTIAL

VCF Operations for Logs Health Check Handbook

Comprehensive Health Verification for VCF Ops for Logs in VCF 9

Prepared by: Virtual Control LLC

Date: March 2026

Version: 1.0

Classification: Internal Use

Platform: VMware Cloud Foundation 9.0

Product: VCF Operations for Logs (formerly VMware Aria Operations for Logs / vRealize Log Insight)

Table of Contents

1. Overview & Purpose

This handbook provides a comprehensive, repeatable methodology for verifying the health of **VCF Operations for Logs** (formerly VMware Aria Operations for Logs / vRealize Log Insight) within a **VMware Cloud Foundation 9** environment. It is designed for infrastructure engineers, VCF administrators, and operations teams who need to validate that the centralized logging platform is functioning correctly, ingesting events at expected rates, and maintaining cluster integrity.

1.1 Health Check Scope

This health check covers the following areas:

- **Service-level health** -- All critical daemons and processes running on each Ops for Logs node
- **Cluster integrity** -- Master/worker node relationships, quorum, and ILB status
- **Storage and retention** -- Disk utilization, Cassandra data footprint, retention and archival policies
- **Ingestion pipeline** -- Event throughput rates, dropped events, and queue depth
- **Log forwarding** -- Outbound syslog/CFAPI forwarding destinations and TLS configuration
- **Content packs** -- Installed packs, versioning, and compatibility with VCF 9
- **Integration health** -- Connectivity between Ops for Logs and VCF Operations (Aria Operations)
- **Agent management** -- Agent connectivity, grouping, and stale agent detection
- **API availability** -- REST API authentication and response times
- **Certificate validity** -- SSL/TLS certificate chain, custom CA, and expiry monitoring
- **Infrastructure services** -- NTP time synchronization and DNS resolution
- **Backup readiness** -- Backup configuration, schedule, and recoverability
- **Resource consumption** -- CPU, memory, disk I/O, and JVM heap on each node

1.2 When to Run This Health Check

Trigger	Frequency	Priority
Scheduled proactive review	Monthly	Standard
Pre-upgrade validation (VCF lifecycle)	Before each upgrade cycle	High
Post-upgrade verification	Immediately after upgrade	Critical
After cluster node addition or removal	As needed	High
After certificate renewal	As needed	High
Performance degradation reported	Reactive	Critical
Ingestion rate anomalies detected	Reactive	Critical
After datacenter-level maintenance window	As needed	Standard
Disaster recovery rehearsal	Quarterly	High

1.3 Component Overview

VCF Operations for Logs in VCF 9 consists of the following architectural components:

Component	Description	Default Port(s)
Log Insight Daemon	Core ingestion and query engine	9000, 9543
Apache HTTPD	Reverse proxy for the web UI and API	443 (HTTPS), 80 (redirect)
Cassandra	Embedded data store for log metadata and indexes	9042, 7000, 7199
Fluentd	Log collection agent framework (embedded)	Various
ILB (Integrated Load Balancer)	Virtual IP distribution across cluster nodes	Same as service ports
REST API	Programmatic access for queries, config, and management	443, 9543
Agents (li-agent)	Remote log collection agents on ESXi and VMs	1514, 514, 6514

Note: In VCF 9, Operations for Logs is deployed and lifecycle-managed through SDDC Manager. The product was previously known as VMware Aria Operations for Logs (8.x) and vRealize Log Insight (pre-8.x). API endpoints and CLI commands remain largely consistent across naming transitions.

2. Prerequisites

2.1 SSH Access

SSH access to each Ops for Logs node is required for service-level and OS-level checks. The default administrative user is `root` or a configured `admin` account.

```
# Test SSH connectivity to each node
ssh root@ops-for-logs-node1.vcf.local "hostname && uptime"
ssh root@ops-for-logs-node2.vcf.local "hostname && uptime"
ssh root@ops-for-logs-node3.vcf.local "hostname && uptime"
```

Expected output:

```
ops-for-logs-node1
 10:23:45 up 45 days,  3:12,  1 user,  load average: 0.42, 0.38, 0.35
```

Warning: If SSH access is disabled or restricted by policy, coordinate with the security team. Many checks in this handbook require shell-level access. API-only alternatives are noted where available.

2.2 API Access & Credentials

All API calls in this handbook target the Ops for Logs REST API at `https://<ops-for-logs-vip>/api/v1/` or `https://<ops-for-logs-vip>/api/v2/`. An authentication token is required for most endpoints.

Obtain an API Session Token

```
# Authenticate and retrieve bearer token
curl -sk -X POST "https://ops-for-logs.vcf.local/api/v1/sessions" \
  -H "Content-Type: application/json" \
  -d '{
    "username": "admin",
    "password": "<ADMIN_PASSWORD>",
    "provider": "Local"
  }'
```

Expected response:

```
{
  "userId": "012345ab-cdef-6789-abcd-ef0123456789",
  "sessionId": "aBcDeFgHiJkLmNoPqRsTuVwXyZ123456",
  "ttl": 1800
}
```

Store the `sessionId` for subsequent API calls:

```
export TOKEN="aBcDeFgHiJkLmNoPqRsTuVwXyZ123456"
```

2.3 Environment Variables

Set these variables at the start of your health check session for convenience:

```
# Ops for Logs VIP or FQDN
export OFL_HOST="ops-for-logs.vcf.local"

# Individual node FQDNs
export OFL_NODE1="ops-for-logs-node1.vcf.local"
export OFL_NODE2="ops-for-logs-node2.vcf.local"
export OFL_NODE3="ops-for-logs-node3.vcf.local"

# API base URL
export OFL_API="https://${OFL_HOST}/api/v1"

# Authenticate and store token
export TOKEN=$(curl -sk -X POST "${OFL_API}/sessions" \
  -H "Content-Type: application/json" \
  -d '{"username":"admin","password":"'"${OFL_PASS}"'","provider":"Local"}' \
  | python3 -c "import sys,json; print(json.load(sys.stdin)['sessionId'])")

echo "Token acquired: ${TOKEN:0:8}..."
```

2.4 Required Tools

Tool	Purpose	Install Check
`curl`	REST API calls	`curl --version`
`jq`	JSON parsing	`jq --version`
`openssl`	Certificate inspection	`openssl version`
`ssh`	Remote node access	`ssh -V`
`python3`	Scripting and JSON parsing	`python3 --version`
`ntpq` / `chronyc`	NTP verification	`ntpq -V` or `chronyc --version`
`dig` / `nslookup`	DNS resolution testing	`dig -v`

3. Quick Reference Summary Table

This table provides a single-glance view of every health check in this handbook, with pass/warn/fail criteria.

#	Check	Command / Method	PASS	WARN	FAIL
4.1	Log Insight Daemon	`systemctl status loginsight`	`active (running)`	Restarting frequently	`inactive` / `failed`
4.2	Cassandra Service	`systemctl status cassandra`	`active (running)`	High compaction pending	`inactive` / `failed`
4.3	Apache HTTPD	`systemctl status httpd`	`active (running)`	High connection count	`inactive` / `failed`
4.4	Fluentd	`systemctl status fluentd`	`active (running)`	Buffer warnings	`inactive` / `failed`
5.1	Node Roles	`GET /api/v1/cluster`	All nodes present	Node degraded	Node missing
5.2	Cluster Status	`GET /api/v1/cluster/status`	All nodes `RUNNING`	Node in `JOINING`	Node `OFFLINE`
5.3	ILB VIP	`curl -sk https://<VIP>/`	HTTP 200/302	High latency (>2s)	Connection refused

6.1	`/storage/var` Usage	`df -h /storage/var`	< 70%	70-85%	> 85%
6.2	Cassandra Data Size	`du -sh /storage/var/cassandra`	< 60% of disk	60-80%	> 80%
7.1	Ingestion Rate	`GET /api/v1/stats`	Stable EPS	> 20% deviation	Ingestion stopped
7.2	Dropped Events	Log analysis	0 dropped	< 0.1% dropped	> 0.1% dropped
8.1	Forwarding Status	`GET /api/v1/forwarding`	All destinations up	Intermittent failures	Destination unreachable
9.1	Content Packs	`GET /api/v1/content/contentpack/list`	All current version	Updates available	Pack errors
10.1	Ops Integration	Launch-in-context test	Works correctly	Partial function	Not configured
11.1	Agent Count	`GET /api/v1/agent/groups`	All agents connected	> 5% stale	> 20% stale
12.1	API Auth	`POST /api/v1/sessions`	Token returned < 2s	Token returned 2-5s	Auth failure
13.1	SSL Certificate	`openssl s_client`	Valid > 30 days	Valid 7-30 days	Expired / < 7 days
14.1	NTP Sync	`chronyc tracking`	Offset < 100ms	Offset 100ms-500ms	Offset > 500ms / unsync
14.2	DNS Resolution	`dig <FQDN>`	Resolves correctly	Slow resolution (>1s)	Resolution fails
15.1	Backup Status	Backup config check	Recent backup exists	Backup > 7 days old	No backup configured
16.1	CPU Utilization	`top` / `mpstat`	< 70% sustained	70-90% sustained	> 90% sustained
16.2	Memory Usage	`free -m`	< 80% used	80-90% used	> 90% used
16.3	JVM Heap	JMX / log analysis	< 75% heap	75-90% heap	> 90% heap / OOM

4. Service Status

All Ops for Logs nodes run a set of critical services. Each must be verified on **every node** in the cluster. Execute the following checks via SSH to each node.

4.1 Log Insight Daemon

The `loginsight` daemon is the core process responsible for log ingestion, indexing, querying, and the web UI.

CLI Check

```
# Check loginsight service status on each node
ssh root@${OFL_NODE1} "systemctl status loginsight"
```

Expected output (healthy):

```
● loginsight.service - VMware Aria Operations for Logs
     Loaded: loaded (/etc/systemd/system/loginsight.service; enabled; vendor preset:
enabled)
     Active: active (running) since Mon 2026-03-20 08:15:22 UTC; 6 days ago
   Main PID: 1842 (loginsight)
      Tasks: 187 (limit: 37253)
     Memory: 4.2G
        CPU: 2d 5h 32min 14.221s
     CGroup: /system.slice/loginsight.service
             └─1842 /usr/lib/loginsight/application/sbin/loginsight ...
```

Uptime and Restart Count Check

```
# Check for recent restarts (indicates instability)
ssh root@${OFL_NODE1} "journalctl -u loginsight --since '7 days ago' | grep -c
'Started VMware'"
```

Expected: `1` (single start in the past 7 days). Values greater than 2 indicate restarts that should be investigated.

Process-Level Verification

```
# Verify the process is running and check resource consumption
ssh root@${OFL_NODE1} "ps aux | grep loginsight | grep -v grep"
```

Criteria	PASS	WARN	FAIL
Service state	`active (running)`	Restarted > 2 times in 7 days	`inactive`, `failed`, or not found
Memory usage	< 80% of allocated	80-90% of allocated	> 90% or OOM killed
Process PID	Stable (same PID for days)	Changed in last 24h	Process not found

Remediation: If the loginsight daemon is not running:
1. Check logs: `journalctl -u loginsight --no-pager -n 100`
2. Check application log: `tail -200 /storage/var/loginsight/runtime.log`
3. Restart the service: `systemctl restart loginsight`
4. If the service fails repeatedly, check disk space on `/storage/var` and Cassandra health.

4.2 Cassandra Service

Cassandra is the embedded database that stores log metadata, indexes, and cluster state. Its health is critical to overall Ops for Logs function.

CLI Check

```
# Check Cassandra service status
ssh root@${OFL_NODE1} "systemctl status cassandra"
```

Expected output (healthy):

```
● cassandra.service - VMware Ops for Logs Cassandra
     Loaded: loaded (/etc/systemd/system/cassandra.service; enabled; vendor preset:
enabled)
     Active: active (running) since Mon 2026-03-20 08:14:55 UTC; 6 days ago
   Main PID: 1523 (java)
      Tasks: 94 (limit: 37253)
     Memory: 2.8G
        CPU: 1d 12h 45min 33.109s
     CGroup: /system.slice/cassandra.service
             └─1523 /usr/bin/java -Xms2048m -Xmx2048m ...
```

Cassandra Node Status (nodetool)

```
# Check Cassandra ring status
ssh root@${OFL_NODE1} "nodetool status"
```

Expected output:

```
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address         Load       Tokens  Owns    Host ID
Rack
UN  192.168.1.101  12.45 GiB  256     33.3%   a1b2c3d4-e5f6-7890-abcd-ef0123456789
rack1
```

```
UN  192.168.1.102  11.82 GiB  256     33.3%   b2c3d4e5-f6a7-8901-bcde-f01234567890
rack1
UN  192.168.1.103  12.01 GiB  256     33.4%   c3d4e5f6-a7b8-9012-cdef-012345678901
rack1
```

The `UN` prefix means **Up** and **Normal**. Any other state requires investigation.

Compaction Check

```
# Check pending compactions
ssh root@${OFL_NODE1} "nodetool compactionstats"
```

Expected: `pending tasks: 0` or a small number (< 10). High pending compactions (> 50) indicate storage I/O pressure.

Criteria	PASS	WARN	FAIL
Service state	`active (running)`	Frequent GC pauses	`inactive` / `failed`
nodetool status	All nodes `UN`	Node in `UJ` (joining)	Node `DN` (down)
Pending compactions	0 - 10	10 - 50	> 50
Data load balance	Within 10% across nodes	10-25% variance	> 25% variance

Remediation: If Cassandra is down or degraded:
1. Check Cassandra logs: `tail -200 /storage/var/cassandra/logs/system.log`
2. Check for heap issues: `grep -i "OutOfMemoryError" /storage/var/cassandra/logs/system.log`
3. Restart Cassandra: `systemctl restart cassandra`
4. If a node shows `DN`, check network connectivity between nodes and verify `/storage/var` has free space.
5. For high compaction backlog, avoid restarting -- allow compaction to complete. Consider increasing compaction throughput: `nodetool setcompactionthroughput 128`

4.3 Apache / HTTPD Service

Apache serves as the reverse proxy for the Ops for Logs web UI and REST API over HTTPS (port 443).

CLI Check

```
# Check Apache HTTPD status
ssh root@${OFL_NODE1} "systemctl status httpd"
```

Expected output (healthy):

```
● httpd.service - The Apache HTTP Server
     Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; vendor preset:
disabled)
     Active: active (running) since Mon 2026-03-20 08:15:30 UTC; 6 days ago
       Docs: man:httpd.service(8)
   Main PID: 2103 (httpd)
     Status: "Total requests: 48231; Idle/Busy workers 8/2"
      Tasks: 213 (limit: 37253)
     Memory: 345.2M
```

Connection Count

```
# Check active connections to port 443
ssh root@${OFL_NODE1} "ss -tuln | grep ':443' && ss -s"
```

Apache Error Log

```
# Check for recent errors
ssh root@${OFL_NODE1} "tail -50 /var/log/httpd/error_log | grep -i 'error\|warn'"
```

Criteria	PASS	WARN	FAIL
Service state	`active (running)`	High worker utilization (> 80%)	`inactive` / `failed`
Port 443 listening	Yes	--	Not listening

Error log	No critical errors	Occasional warnings	Persistent errors

Remediation: If Apache is down:
1. Check config syntax: `httpd -t`
2. Check error log: `tail -100 /var/log/httpd/error_log`
3. Verify SSL certificate files exist and are readable
4. Restart: `systemctl restart httpd`

4.4 Fluentd Service

Fluentd handles local log collection and forwarding on each node.

CLI Check

```
# Check Fluentd service status
ssh root@${OFL_NODE1} "systemctl status fluentd"
```

Expected output (healthy):

```
● fluentd.service - Fluentd Log Collector
     Loaded: loaded (/etc/systemd/system/fluentd.service; enabled; vendor preset:
enabled)
     Active: active (running) since Mon 2026-03-20 08:15:25 UTC; 6 days ago
   Main PID: 1955 (ruby)
      Tasks: 18 (limit: 37253)
     Memory: 128.5M
```

Buffer Health

```
# Check Fluentd buffer directory size
ssh root@${OFL_NODE1} "du -sh /storage/var/fluentd/buffer/ 2>/dev/null || echo
'No buffer directory'"

# Check for buffer overflow warnings in Fluentd logs
ssh root@${OFL_NODE1} "grep -c 'buffer is full' /var/log/fluentd/fluentd.log
2>/dev/null || echo '0'"
```

Criteria	PASS	WARN	FAIL
Service state	`active (running)`	Buffer warnings present	`inactive` / `failed`
Buffer size	< 100 MB	100 MB - 500 MB	> 500 MB (backlog)
Buffer overflow events	0	1-5 in past 24h	> 5 in past 24h

Remediation: If Fluentd has buffer issues:
1. Check log: `tail -100 /var/log/fluentd/fluentd.log`

2. Clear stale buffers (if safe): `rm -f /storage/var/fluentd/buffer/*.log`
3. Restart: `systemctl restart fluentd`
4. Investigate downstream destination availability if buffers are growing.

4.5 All Services Summary Check

Run this consolidated command on each node to verify all critical services in a single pass:

```
# Quick service health summary for a single node
ssh root@${OFL_NODE1} 'echo "=== Service Status Summary ===" && \
  for svc in loginsight cassandra httpd fluentd; do \
    STATUS=$(systemctl is-active $svc 2>/dev/null); \
    ENABLED=$(systemctl is-enabled $svc 2>/dev/null); \
    printf "%-15s Active: %-12s Enabled: %s\n" "$svc" "$STATUS" "$ENABLED"; \
  done'
```

Expected output:

```
=== Service Status Summary ===
loginsight      Active: active       Enabled: enabled
cassandra       Active: active       Enabled: enabled
httpd           Active: active       Enabled: enabled
fluentd         Active: active       Enabled: enabled
```

Check All Nodes at Once

```
# Loop across all cluster nodes
for NODE in ${OFL_NODE1} ${OFL_NODE2} ${OFL_NODE3}; do
  echo "===== ${NODE} ====="
  ssh root@${NODE} 'for svc in loginsight cassandra httpd fluentd; do \
    printf "%-15s %s\n" "$svc" "$(systemctl is-active $svc)"; done'
  echo ""
done
```

5. Cluster Health

VCF Operations for Logs operates as a clustered appliance with a minimum of three nodes for high availability. Cluster health verification ensures that all nodes are online, roles are correctly assigned, and the integrated load balancer is distributing traffic.

5.1 Node Roles (Master / Worker)

Each Ops for Logs cluster has exactly one **master** node and one or more **worker** nodes. The master manages cluster coordination, schema, and configuration replication.

API Check

```
# Retrieve cluster node roles
curl -sk -X GET "${OFL_API}/cluster" \
  -H "Authorization: Bearer ${TOKEN}" | jq '.'
```

Expected response:


```
{
  "clusterSize": 3,
  "nodes": [
    {
      "id": "a1b2c3d4-e5f6-7890-abcd-ef0123456789",
      "hostname": "ops-for-logs-node1.vcf.local",
      "ipAddress": "192.168.1.101",
      "role": "MASTER",
      "status": "RUNNING",
      "version": "9.0.0-12345678"
    },
    {
      "id": "b2c3d4e5-f6a7-8901-bcde-f01234567890",
      "hostname": "ops-for-logs-node2.vcf.local",
      "ipAddress": "192.168.1.102",
      "role": "WORKER",
      "status": "RUNNING",
      "version": "9.0.0-12345678"
    },
    {
      "id": "c3d4e5f6-a7b8-9012-cdef-012345678901",
      "hostname": "ops-for-logs-node3.vcf.local",
      "ipAddress": "192.168.1.103",
      "role": "WORKER",
      "status": "RUNNING",
      "version": "9.0.0-12345678"
    }
```

```
    ]
}
```

Validation Criteria

Criteria	PASS	WARN	FAIL
Master node present	Exactly 1 master	--	0 or > 1 master
All nodes reporting	Count matches `clusterSize`	--	Missing node(s)
Version consistency	All nodes same version	--	Version mismatch
All nodes RUNNING	All status = `RUNNING`	Node in `JOINING` / `LEAVING`	Node `OFFLINE` / `ERROR`

Remediation: If a node is missing or offline:
1. SSH to the affected node and check `systemctl status loginsight`
2. Check network connectivity: `ping ${OFL_NODE1}` from other nodes
3. Verify the node can reach the master on port 9000: `curl -sk https://${OFL_NODE1}:9000`
4. Review cluster join logs: `tail -200 /storage/var/loginsight/runtime.log | grep -i cluster`
5. If a node is stuck in JOINING, it may need to be removed and re-added via the admin UI.

5.2 Cluster Status via API

Detailed Cluster Status

```
# Get detailed cluster health
curl -sk -X GET "${OFL_API}/cluster/status" \
  -H "Authorization: Bearer ${TOKEN}" | jq '.'
```

Expected response:


```
{
  "clusterStatus": "RUNNING",
  "masterNodeId": "a1b2c3d4-e5f6-7890-abcd-ef0123456789",
  "nodesHealth": [
    {
      "nodeId": "a1b2c3d4-e5f6-7890-abcd-ef0123456789",
      "hostname": "ops-for-logs-node1.vcf.local",
      "state": "RUNNING",
      "diskUsagePercent": 42.5,
      "cpuUsagePercent": 23.1,
      "memoryUsagePercent": 65.8,
      "eventsPerSecond": 3245
    },
    {
      "nodeId": "b2c3d4e5-f6a7-8901-bcde-f01234567890",
      "hostname": "ops-for-logs-node2.vcf.local",
      "state": "RUNNING",
      "diskUsagePercent": 41.2,
      "cpuUsagePercent": 21.8,
      "memoryUsagePercent": 63.4,
      "eventsPerSecond": 3198
    },
    {
      "nodeId": "c3d4e5f6-a7b8-9012-cdef-012345678901",
      "hostname": "ops-for-logs-node3.vcf.local",
      "state": "RUNNING",
      "diskUsagePercent": 43.1,
      "cpuUsagePercent": 22.5,
      "memoryUsagePercent": 64.2,
```

```
      "eventsPerSecond": 3210
    }
  ]
}
```

5.3 Integrated Load Balancer (ILB)

The ILB provides a single virtual IP (VIP) that distributes incoming log traffic and API requests across all cluster nodes.

VIP Reachability

```
# Test VIP is responding
curl -sk -o /dev/null -w "HTTP_CODE: %{http_code}\nTIME_TOTAL: %{time_total}s\n"
\
  "https://${OFL_HOST}/"
```

Expected output:

```
HTTP_CODE: 302
TIME_TOTAL: 0.234s
```

ILB Configuration via API

```
# Check ILB configuration
curl -sk -X GET "${OFL_API}/ilb" \
  -H "Authorization: Bearer ${TOKEN}" | jq '.'
```

Expected response:

```
{
  "enabled": true,
  "virtualIp": "192.168.1.100",
  "heartbeatInterval": 3,
  "failoverTimeout": 15
}
```

Criteria	PASS	WARN	FAIL
VIP responds	HTTP 200 or 302	Response time > 2s	Connection refused / timeout
ILB enabled	`true`	--	`false`

All nodes behind ILB	All nodes included	--	Node excluded

Remediation: If the VIP is unreachable:
1. Check if the VIP is bound to a node: `ssh root@${OFL_NODE1} "ip addr show | grep 192.168.1.100"`
2. Verify ILB is enabled in the admin UI under **Administration > Cluster > ILB**
3. Check for IP conflicts with `arping -D -I eth0 192.168.1.100`
4. Restart ILB by restarting the loginsight service on the master node.

5.4 Node-to-Node Connectivity

All cluster nodes must be able to communicate with each other on required ports.

```
# Test connectivity from node1 to node2 and node3 on key ports
ssh root@${OFL_NODE1} "
  echo '--- Port 9000 (loginsight) ---'
  nc -zv ${OFL_NODE2} 9000 2>&1
  nc -zv ${OFL_NODE3} 9000 2>&1
  echo '--- Port 9042 (Cassandra CQL) ---'
  nc -zv ${OFL_NODE2} 9042 2>&1
  nc -zv ${OFL_NODE3} 9042 2>&1
  echo '--- Port 7000 (Cassandra inter-node) ---'
  nc -zv ${OFL_NODE2} 7000 2>&1
  nc -zv ${OFL_NODE3} 7000 2>&1
"
```

Expected output:

```
--- Port 9000 (loginsight) ---
Connection to ops-for-logs-node2.vcf.local 9000 port [tcp/*] succeeded!
Connection to ops-for-logs-node3.vcf.local 9000 port [tcp/*] succeeded!
--- Port 9042 (Cassandra CQL) ---
Connection to ops-for-logs-node2.vcf.local 9042 port [tcp/*] succeeded!
Connection to ops-for-logs-node3.vcf.local 9042 port [tcp/*] succeeded!
--- Port 7000 (Cassandra inter-node) ---
Connection to ops-for-logs-node2.vcf.local 7000 port [tcp/*] succeeded!
Connection to ops-for-logs-node3.vcf.local 7000 port [tcp/*] succeeded!
```

6. Disk & Storage Health

Storage is the most common source of Ops for Logs issues. The appliance stores all ingested log data, Cassandra metadata, and indexes on the `/storage/var` partition.

6.1 Storage Partition Layout

Check Disk Layout

```
# Show all mounted partitions and usage
ssh root@${OFL_NODE1} "df -hT"
```

Expected output:

```
Filesystem     Type      Size  Used Avail Use% Mounted on
/dev/sda3      ext4       10G  3.2G  6.3G  34% /
/dev/sda1      vfat      512M   12M  500M   3% /boot/efi
/dev/sdb1      ext4      500G  210G  266G  45% /storage/var
tmpfs          tmpfs     7.8G     0  7.8G   0% /dev/shm
```

The critical partitions are:

Partition	Purpose	Minimum Size	Alert Threshold
`/`	OS root filesystem	10 GB	> 80% used
`/storage/var`	Log data, Cassandra, indexes	500 GB+	> 70% used
`/boot/efi`	EFI boot partition	512 MB	> 90% used

6.2 Storage Usage Thresholds

Detailed Storage Check

```
# Check /storage/var utilization with breakdown
ssh root@${OFL_NODE1} "
  echo '=== Overall /storage/var ==='
  df -h /storage/var
  echo ''
  echo '=== Top-level directories by size ==='
  du -sh /storage/var/*/ 2>/dev/null | sort -rh | head -20
"
```

Expected output:

```
=== Overall /storage/var ===
Filesystem      Size  Used Avail Use% Mounted on
/dev/sdb1       500G  210G  266G  45% /storage/var

=== Top-level directories by size ===
185G    /storage/var/loginsight/
18G     /storage/var/cassandra/
3.2G    /storage/var/fluentd/
1.1G    /storage/var/apache/
```

Criteria	PASS	WARN	FAIL
`/storage/var` usage	< 70%	70-85%	> 85%
Root `/` usage	< 80%	80-90%	> 90%
Inode usage	< 70%	70-85%	> 85%

Inode Check

```
# Check inode usage (often overlooked)
ssh root@${OFL_NODE1} "df -i /storage/var"
```

> **Warning:** When `/storage/var` exceeds 85%, Ops for Logs will begin aggressively purging old data. At 95%, ingestion may halt entirely. Proactive monitoring is essential.

6.3 Cassandra Data Size

```
# Check Cassandra data footprint
ssh root@${OFL_NODE1} "
  echo '=== Cassandra Data Directory ==='
  du -sh /storage/var/cassandra/data/ 2>/dev/null
  echo ''
  echo '=== Cassandra Commit Logs ==='
  du -sh /storage/var/cassandra/commitlog/ 2>/dev/null
  echo ''
  echo '=== Cassandra Saved Caches ==='
  du -sh /storage/var/cassandra/saved_caches/ 2>/dev/null
"
```

Criteria	PASS	WARN	FAIL
Data directory	< 60% of `/storage/var`	60-80%	> 80%
Commit log size	< 2 GB	2-5 GB	> 5 GB (indicates write issues)

6.4 Retention Policy

Check Retention Configuration via API

```
# Get retention settings
curl -sk -X GET "${OFL_API}/time/config" \
  -H "Authorization: Bearer ${TOKEN}" | jq '.'
```

Expected response:

```
{
  "retentionPeriod": 30,
  "archiveEnabled": true,
  "archiveRetentionPeriod": 365
}
```

Check Retention via CLI

```
# Check the loginsight configuration file for retention settings
ssh root@${OFL_NODE1} "grep -i 'retention'
/storage/var/loginsight/config/loginsight-config.xml 2>/dev/null"
```

6.5 Archive Configuration

```
# Check archive/NFS configuration
curl -sk -X GET "${OFL_API}/archive" \
  -H "Authorization: Bearer ${TOKEN}" | jq '.'
```

Expected response (when configured):

```
{
  "enabled": true,
  "archiveType": "NFS",
  "nfsServer": "nfs-server.vcf.local",
  "nfsPath": "/exports/loginsight-archive",
  "archiveFrequency": "DAILY",
  "compressionEnabled": true
}
```

Verify Archive Mount

```
# Check if NFS archive is mounted
ssh root@${OFL_NODE1} "mount | grep nfs && df -h
/storage/var/loginsight/archive/"
```

Remediation: If storage is critically low:
1. Reduce retention period via API: reduce `retentionPeriod` value
2. Enable archiving to offload old data to NFS
3. Expand the `/storage/var` virtual disk in vSphere and grow the filesystem
4. Check for and remove stale Cassandra snapshots: `nodetool clearsnapshot`

7. Ingestion Rate Monitoring

The ingestion rate (events per second, or EPS) is a key performance indicator for Ops for Logs. Monitoring this metric ensures that the platform is receiving logs at expected volumes and not silently dropping events.

7.1 Events Per Second

API Check

```
# Get current ingestion statistics
curl -sk -X GET "${OFL_API}/stats" \
  -H "Authorization: Bearer ${TOKEN}" | jq '.'
```

Expected response:

```
{
  "totalEventsIngested": 285432109,
  "currentEventsPerSecond": 9653,
  "averageEventsPerSecond": 9480,
  "peakEventsPerSecond": 18234,
  "totalBytesIngested": 412983726501,
  "droppedEvents": 0,
  "queueDepth": 12
}
```

CLI-Based Ingestion Monitoring

```
# Monitor real-time ingestion rate from node logs
ssh root@${OFL_NODE1} "tail -100 /storage/var/loginsight/runtime.log | grep -i
'ingestion\|eps\|events.*second'"
```

Historical Ingestion Query

```
# Query ingestion rate over the past 24 hours
curl -sk -X POST "${OFL_API}/events/stats" \
  -H "Authorization: Bearer ${TOKEN}" \
  -H "Content-Type: application/json" \
  -d '{
    "query": "ingestion_rate",
```

```
    "startTimeMillis": '$(date -d "24 hours ago" +%s%3N)',
    "endTimeMillis": '$(date +%s%3N)',
    "bucketDurationMinutes": 60
  }' | jq '.buckets[] | {time: .startTime, eps: .eventsPerSecond}'
```

Criteria	PASS	WARN	FAIL
Current EPS	Within 20% of baseline	20-50% deviation from baseline	> 50% deviation or 0 EPS
Dropped events	0	< 0.1% of total ingested	> 0.1% of total
Queue depth	< 100	100-1000	> 1000

7.2 Ingestion Pipeline Health

```
# Check ingestion pipeline components
ssh root@${OFL_NODE1} "
  echo '=== Listening Ports for Ingestion ==='
  ss -tuln | grep -E ':(514|1514|6514|9000|9543) '
  echo ''
  echo '=== Active Syslog Connections ==='
  ss -tn | grep -E ':(514|1514|6514) ' | wc -l
  echo ''
  echo '=== Active CFAPI Connections ==='
  ss -tn | grep -E ':(9000|9543) ' | wc -l
"
```

Expected output:

```
=== Listening Ports for Ingestion ===
tcp   LISTEN  0  128  *:514    *:*
tcp   LISTEN  0  128  *:1514   *:*
tcp   LISTEN  0  128  *:6514   *:*
tcp   LISTEN  0  128  *:9000   *:*
tcp   LISTEN  0  128  *:9543   *:*

=== Active Syslog Connections ===
42

=== Active CFAPI Connections ===
18
```

7.3 Dropped Events & Queue Depth

```
# Check for dropped events in the runtime log
ssh root@${OFL_NODE1} "grep -c 'dropped\|overflow\|backpressure' \
  /storage/var/loginsight/runtime.log 2>/dev/null || echo '0'"

# Check ingestion queue depth
ssh root@${OFL_NODE1} "grep -i 'queue.*depth\|pending.*events' \
  /storage/var/loginsight/runtime.log | tail -5"
```

Remediation: If ingestion is dropping events:
1. Check disk space -- full storage is the most common cause
2. Review Cassandra health -- Cassandra write failures block ingestion
3. Check for network saturation on ingestion ports
4. Scale out by adding worker nodes if sustained EPS exceeds capacity
5. Review forwarding destinations -- slow downstream targets can cause backpressure

8. Log Forwarding Configuration

Ops for Logs can forward ingested logs to external destinations via syslog (UDP/TCP), syslog over TLS, or the CFAPI protocol. This section verifies that all forwarding destinations are configured correctly and operating.

8.1 Forwarding Destinations

API Check

```
# List all configured forwarding destinations
curl -sk -X GET "${OFL_API}/forwarding" \
  -H "Authorization: Bearer ${TOKEN}" | jq '.'
```

Expected response:


```
{
  "destinations": [
    {
      "id": "dest-001",
      "name": "SIEM-Primary",
      "host": "siem.vcf.local",
      "port": 6514,
      "protocol": "SYSLOG",
      "transport": "TCP-TLS",
      "enabled": true,
      "status": "CONNECTED",
      "filter": "*",
      "lastEventForwarded": "2026-03-26T09:45:12Z"
    },
    {
      "id": "dest-002",
      "name": "Archive-Collector",
      "host": "log-archive.vcf.local",
      "port": 9543,
      "protocol": "CFAPI",
      "transport": "HTTPS",
      "enabled": true,
      "status": "CONNECTED",
      "filter": "vmw_vc_*",
      "lastEventForwarded": "2026-03-26T09:45:10Z"
    }
  ]
}
```

8.2 Protocol & TLS Configuration

Verify TLS Configuration for Syslog Forwarding

```
# Check TLS certificate used for syslog forwarding
ssh root@${OFL_NODE1} "
  echo '=== Forwarding TLS Certificates ==='
  ls -la /storage/var/loginsight/certs/forwarding/ 2>/dev/null || echo 'No
forwarding certs directory'
  echo ''
  echo '=== Forwarding Configuration ==='
  grep -A 10 'forwarding' /storage/var/loginsight/config/loginsight-config.xml
2>/dev/null | head -30
"
```

Test TLS Connectivity to Forwarding Destination

```
# Verify TLS handshake to syslog destination
openssl s_client -connect siem.vcf.local:6514 -servername siem.vcf.local
</dev/null 2>/dev/null | \
  openssl x509 -noout -subject -dates -issuer
```

Expected output:

```
subject=CN = siem.vcf.local
notBefore=Jan 15 00:00:00 2026 GMT
notAfter=Jan 15 23:59:59 2027 GMT
issuer=CN = VCF Internal CA, O = Virtual Control LLC
```

Criteria	PASS	WARN	FAIL
TLS handshake	Succeeds	Certificate nearing expiry	Handshake fails
Protocol match	Matches destination config	--	Mismatch
Certificate trust	CA chain trusted	Self-signed (intentional)	Untrusted / expired

8.3 Forwarding Health Verification

```
# Check forwarding statistics per destination
curl -sk -X GET "${OFL_API}/forwarding/stats" \
  -H "Authorization: Bearer ${TOKEN}" | jq '.destinations[] | {name,
eventsForwarded, eventsFailed, lastSuccess}'
```

Expected output:

```
{
  "name": "SIEM-Primary",
  "eventsForwarded": 48293012,
  "eventsFailed": 0,
  "lastSuccess": "2026-03-26T09:45:12Z"
}
{
  "name": "Archive-Collector",
  "eventsForwarded": 12045231,
  "eventsFailed": 0,
  "lastSuccess": "2026-03-26T09:45:10Z"
}
```

Criteria	PASS	WARN	FAIL
Events forwarded	Increasing steadily	Intermittent pauses	Not increasing / 0
Events failed	0	< 0.01% of forwarded	> 0.01% or increasing
Last success	Within 5 minutes	5-60 minutes ago	> 60 minutes ago
Destination status	`CONNECTED`	`RECONNECTING`	`DISCONNECTED` / `ERROR`

8.4 Test Forwarding

```
# Send a test event via the API to verify end-to-end forwarding
curl -sk -X POST "${OFL_API}/events/ingest/0" \
  -H "Authorization: Bearer ${TOKEN}" \
  -H "Content-Type: application/json" \
  -d '{
    "events": [
      {
        "text": "HEALTH_CHECK_TEST: Forwarding validation event from Ops for Logs
health check - '"$(date -u +%Y-%m-%dT%H:%M:%SZ)"'",
        "source": "health-check-script",
        "fields": [
          {"name": "test_id", "content": "hc-fwd-'"$(date +%s)"'"}
        ]
      }
    ]
  }'
```

Then verify the test event arrived at the forwarding destination by searching for `HEALTH_CHECK_TEST` in the target SIEM or log collector.

Remediation: If forwarding is failing:
1. Verify destination reachability: `nc -zv siem.vcf.local 6514`
2. Check firewall rules between Ops for Logs nodes and the destination
3. Verify TLS certificate compatibility -- the destination must trust the Ops for Logs CA
4. Restart forwarding by toggling the destination off and on via the UI
5. Check destination-side logs for connection rejections

9. Content Packs

Content packs provide pre-built dashboards, alerts, extracted fields, and queries for specific products (vSphere, NSX, SDDC Manager, vSAN, etc.). Keeping content packs current ensures full observability.

9.1 Installed Content Packs

API Check

```
# List all installed content packs
curl -sk -X GET "${OFL_API}/content/contentpack/list" \
  -H "Authorization: Bearer ${TOKEN}" | jq '.contentPacks[] | {name, namespace,
version, installedDate}'
```

Expected output:


```
{
  "name": "VMware vSphere",
  "namespace": "com.vmware.vsphere",
  "version": "9.0.1",
  "installedDate": "2026-02-15T10:30:00Z"
}
{
  "name": "VMware NSX",
  "namespace": "com.vmware.nsx",
  "version": "9.0.0",
  "installedDate": "2026-02-15T10:30:05Z"
}
{
  "name": "VMware SDDC Manager",
  "namespace": "com.vmware.sddc",
  "version": "9.0.0",
  "installedDate": "2026-02-15T10:30:10Z"
}
{
  "name": "VMware vSAN",
  "namespace": "com.vmware.vsan",
  "version": "9.0.0",
  "installedDate": "2026-02-15T10:30:15Z"
}
{
  "name": "VMware Aria Operations",
  "namespace": "com.vmware.vrops",
  "version": "9.0.0",
```

```
  "installedDate": "2026-02-15T10:30:20Z"
}
```

Essential Content Packs for VCF 9

Content Pack	Namespace	Minimum Version	Purpose
VMware vSphere	`com.vmware.vsphere`	9.0.0	ESXi and vCenter log parsing
VMware NSX	`com.vmware.nsx`	9.0.0	NSX manager and edge log parsing
VMware SDDC Manager	`com.vmware.sddc`	9.0.0	SDDC Manager lifecycle events
VMware vSAN	`com.vmware.vsan`	9.0.0	vSAN health and performance logs
VMware Aria Operations	`com.vmware.vrops`	9.0.0	Ops manager integration logs
Linux	`com.vmware.linux`	9.0.0	General Linux syslog parsing
General	`com.vmware.general`	9.0.0	Generic field extraction

9.2 Version Status & Updates

Check for Available Updates

```
# Check marketplace for content pack updates
curl -sk -X GET "${OFL_API}/content/contentpack/marketplace" \
  -H "Authorization: Bearer ${TOKEN}" | jq '.contentPacks[] |
select(.updateAvailable == true) | {name, currentVersion, availableVersion}'
```

Expected output (no updates needed):

```
(empty output -- no updates available)
```

Output when updates are available:

```
{
  "name": "VMware vSphere",
  "currentVersion": "9.0.0",
  "availableVersion": "9.0.1"
}
```

Criteria	PASS	WARN	FAIL
All VCF packs installed	All 7+ packs present	Missing non-critical pack	Missing vSphere or SDDC pack
Pack versions	All at latest	Minor update available	Major version behind
Pack status	No errors	Warning on extraction	Pack failed to load

9.3 Auto-Update Configuration

```
# Check auto-update settings for content packs
curl -sk -X GET "${OFL_API}/content/contentpack/autoupdate" \
  -H "Authorization: Bearer ${TOKEN}" | jq '.'
```

Expected response:

```
{
  "autoUpdateEnabled": true,
  "checkIntervalHours": 24,
  "lastCheckTime": "2026-03-25T02:00:00Z",
  "proxyEnabled": false
}
```

Remediation: If content packs are outdated or missing:
1. Update individual pack: Navigate to **Content Packs** in the UI, select the pack, click **Update**
2. Install missing pack via API: `POST /api/v1/content/contentpack/install` with the pack namespace
3. If marketplace is unreachable, download packs manually from the VMware Marketplace and upload via UI
4. Enable auto-update: `PUT /api/v1/content/contentpack/autoupdate` with `{"autoUpdateEnabled": true}`

10. Integration with VCF Operations

VCF Operations for Logs integrates with VCF Operations (formerly Aria Operations / vRealize Operations) to provide launch-in-context capabilities, shared authentication, and correlated alerting.

10.1 Launch-in-Context Configuration

Launch-in-context enables users to jump directly from VCF Operations alerts and dashboards into relevant log queries in Ops for Logs.

Verify Integration Configuration

```
# Check VCF Operations integration settings
curl -sk -X GET "${OFL_API}/integration/vrops" \
  -H "Authorization: Bearer ${TOKEN}" | jq '.'
```

Expected response:

```
{
  "enabled": true,
  "vropsHost": "ops.vcf.local",
  "vropsPort": 443,
  "connectionStatus": "CONNECTED",
  "lastSyncTime": "2026-03-26T08:00:00Z",
  "ssoIntegrated": true,
  "launchInContextEnabled": true
}
```

Test Launch-in-Context URL Generation

```
# Verify launch-in-context URL format
curl -sk -X GET "${OFL_API}/integration/vrops/launch-url?resourceId=vm-
123&timeRange=3600" \
  -H "Authorization: Bearer ${TOKEN}" | jq '.'
```

Criteria	PASS	WARN	FAIL
Integration enabled	`true`	--	`false` or not configured
Connection status	`CONNECTED`	`DEGRADED`	`DISCONNECTED`
Last sync time	Within 24 hours	1-7 days ago	> 7 days or never

Launch-in-context	URL generated correctly	Partial functionality	Errors on generation

10.2 Shared Authentication

```
# Verify SSO / shared authentication with VCF Operations
curl -sk -X GET "${OFL_API}/auth/providers" \
  -H "Authorization: Bearer ${TOKEN}" | jq '.'
```

Expected response:

```
{
  "providers": [
    {
      "name": "Local",
      "type": "LOCAL",
      "enabled": true
    },
    {
      "name": "vcf-sso.vcf.local",
      "type": "ACTIVE_DIRECTORY",
      "enabled": true,
      "connectionStatus": "CONNECTED"
    },
    {
      "name": "VMware Identity Manager",
      "type": "VIDM",
      "enabled": true,
      "connectionStatus": "CONNECTED"
    }
  ]
}
```

Criteria	PASS	WARN	FAIL
SSO provider configured	Yes, `CONNECTED`	Configured but `DEGRADED`	Not configured
AD integration	`CONNECTED`	Intermittent failures	`DISCONNECTED`
Local auth backup	Enabled as fallback	--	Disabled (no fallback)

10.3 Data Flow Verification

Verify that VCF Operations is sending notification events and that Ops for Logs is receiving them.

```
# Search for VCF Operations events in Ops for Logs
curl -sk -X POST "${OFL_API}/events" \
  -H "Authorization: Bearer ${TOKEN}" \
  -H "Content-Type: application/json" \
  -d '{
    "query": "vmw_product=vrops",
    "startTimeMillis": '$(date -d "24 hours ago" +%s%3N)',
    "endTimeMillis": '$(date +%s%3N)',
    "limit": 5
  }' | jq '.results | length'
```

Expected: A positive number indicating events are flowing from VCF Operations to Ops for Logs.

Remediation: If integration is broken:
1. Re-register the integration from VCF Operations: **Administration > Management > Log Insight Integration**
2. Verify network connectivity: `curl -sk https://ops.vcf.local:443` from Ops for Logs nodes
3. Check SSO token validity -- re-authenticate if tokens have expired
4. Verify VIDM (Workspace ONE Access) is operational if using VIDM-based SSO
5. Restart the integration service: `systemctl restart loginsight` (integration is part of the main daemon)

11. Agent Status

Ops for Logs agents (li-agent) run on ESXi hosts, VMs, and other endpoints to collect and forward logs to the cluster. Agent health monitoring ensures complete log coverage.

11.1 Connected Agents

API Check

```
# Get agent summary statistics
curl -sk -X GET "${OFL_API}/agent/stats" \
  -H "Authorization: Bearer ${TOKEN}" | jq '.'
```

Expected response:

```
{
  "totalAgents": 48,
  "connectedAgents": 47,
  "disconnectedAgents": 1,
  "activeAgentGroups": 5,
  "averageEventsPerAgent": 201
}
```

List All Connected Agents

```
# List agents with their connection status
curl -sk -X GET "${OFL_API}/agent/agents" \
  -H "Authorization: Bearer ${TOKEN}" | jq '.agents[] | {hostname, ipAddress,
version, status, lastHeartbeat}' | head -60
```

Sample output:

```
{
  "hostname": "esxi-host-01.vcf.local",
  "ipAddress": "192.168.10.101",
  "version": "9.0.0-12345",
  "status": "CONNECTED",
  "lastHeartbeat": "2026-03-26T09:44:55Z"
}
{
```

```
  "hostname": "esxi-host-02.vcf.local",
  "ipAddress": "192.168.10.102",
  "version": "9.0.0-12345",
  "status": "CONNECTED",
  "lastHeartbeat": "2026-03-26T09:44:52Z"
}
```

Criteria	PASS	WARN	FAIL
Connected agents	100% connected	95-99% connected	< 95% connected
Agent version	All same version as cluster	Minor version mismatch	Major version mismatch
Heartbeat age	< 5 minutes	5-30 minutes	> 30 minutes

11.2 Agent Groups

Agent groups organize agents for targeted log collection and configuration distribution.

```
# List all agent groups
curl -sk -X GET "${OFL_API}/agent/groups" \
  -H "Authorization: Bearer ${TOKEN}" | jq '.groups[] | {id, name, agentCount,
filter}'
```

Expected output:

```
{
  "id": "group-001",
  "name": "ESXi-Hosts",
  "agentCount": 32,
  "filter": "hostname MATCHES esxi-*"
}
{
  "id": "group-002",
  "name": "VCF-Management-VMs",
  "agentCount": 12,
  "filter": "hostname MATCHES vcf-mgmt-*"
}
{
  "id": "group-003",
  "name": "Windows-Servers",
  "agentCount": 4,
  "filter": "os MATCHES Windows*"
}
```

Verify Agent Group Configuration

```
# Get detailed agent group configuration including collection targets
curl -sk -X GET "${OFL_API}/agent/groups/group-001" \
  -H "Authorization: Bearer ${TOKEN}" | jq '.'
```

Expected response:

```
{
  "id": "group-001",
  "name": "ESXi-Hosts",
  "agentCount": 32,
  "config": {
    "fileLogs": [
      {
        "directory": "/var/log",
        "include": "*.log",
        "parser": "AUTO"
      },
      {
        "directory": "/var/run/log",
        "include": "vmkernel*",
        "parser": "VMW_ESXI"
      }
    ],
    "eventLogs": [],
    "destination": {
      "host": "ops-for-logs.vcf.local",
      "port": 9543,
      "protocol": "CFAPI",
      "ssl": true
    }
  }
}
```

11.3 Stale Agent Detection

Stale agents are agents that have not sent a heartbeat within the expected interval (typically 5 minutes). They may indicate agent crashes, network issues, or decommissioned hosts.

```
# Find agents with no heartbeat in the last 30 minutes
curl -sk -X GET "${OFL_API}/agent/agents?status=DISCONNECTED" \
  -H "Authorization: Bearer ${TOKEN}" | jq '.agents[] | {hostname, lastHeartbeat, status}'
```

Expected output (ideally empty):

```
{
  "hostname": "old-vm-decommissioned.vcf.local",
  "lastHeartbeat": "2026-03-10T14:22:00Z",
  "status": "DISCONNECTED"
}
```

Remediation: For stale/disconnected agents:
1. Verify the host is still operational: `ping old-vm-decommissioned.vcf.local`
2. If the host is active, SSH in and check agent status: `systemctl status liagentd`
3. Restart the agent: `systemctl restart liagentd`
4. Check agent logs: `tail -100 /var/log/liagent/liagent.log`
5. For decommissioned hosts, remove the stale agent entry via API: `DELETE /api/v1/agent/agents/{agentId}`
6. Verify agent can reach Ops for Logs on port 9543: `nc -zv ops-for-logs.vcf.local 9543`

12. API Health

The Ops for Logs REST API is the primary interface for programmatic queries, configuration management, and integration with external tools. Verifying API health ensures automation and integrations function correctly.

12.1 Token Acquisition

Timed Authentication Test

```
# Measure authentication response time
time curl -sk -X POST "${OFL_API}/sessions" \
  -H "Content-Type: application/json" \
  -d '{"username":"admin","password":"'"${OFL_PASS}"'","provider":"Local"}' \
  -o /dev/null -w "HTTP_CODE: %{http_code}\nTIME_TOTAL: %
{time_total}s\nTIME_CONNECT: %{time_connect}s\n"
```

Expected output:

```
HTTP_CODE: 200
TIME_TOTAL: 0.345s
TIME_CONNECT: 0.012s

real    0m0.362s
user    0m0.024s
sys     0m0.012s
```

Test Token Validity

```
# Verify a token works for an authenticated endpoint
curl -sk -X GET "${OFL_API}/version" \
  -H "Authorization: Bearer ${TOKEN}" \
  -w "\nHTTP_CODE: %{http_code}\n" | jq '.'
```

Expected response:

```
{
  "version": "9.0.0",
  "build": "12345678",
  "releaseName": "VCF Operations for Logs 9.0"
```

```
}
HTTP_CODE: 200
```

Test Invalid Token Handling

```
# Verify that invalid tokens are properly rejected
curl -sk -X GET "${OFL_API}/cluster" \
  -H "Authorization: Bearer INVALID_TOKEN_12345" \
  -w "\nHTTP_CODE: %{http_code}\n"
```

Expected: `HTTP_CODE: 401` (Unauthorized).

Criteria	PASS	WARN	FAIL
Auth response time	< 2 seconds	2-5 seconds	> 5 seconds or timeout
HTTP status	200	--	401, 403, 500, or connection error
Token validity	Token works on subsequent calls	TTL shorter than expected	Token immediately invalid
Invalid token rejection	Returns 401	--	Returns 200 (security issue)

12.2 API Responsiveness

Test several key API endpoints for response time under normal load.

```
# Benchmark multiple API endpoints
echo "=== API Endpoint Response Times ==="
for ENDPOINT in "version" "cluster" "cluster/status" "stats" "agent/stats"
"forwarding"; do
  RESP=$(curl -sk -X GET "${OFL_API}/${ENDPOINT}" \
    -H "Authorization: Bearer ${TOKEN}" \
    -o /dev/null -w "%{http_code} %{time_total}s")
  printf "%-25s %s\n" "${ENDPOINT}" "${RESP}"
done
```

Expected output:

```
=== API Endpoint Response Times ===
version                   200 0.089s
cluster                   200 0.156s
cluster/status            200 0.234s
stats                     200 0.312s
agent/stats               200 0.198s
forwarding                200 0.145s
```

Criteria	PASS	WARN	FAIL
Average response time	< 1 second	1-3 seconds	> 3 seconds
All endpoints reachable	All return 200	Some return 503	Critical endpoints fail
Error rate	0%	< 1%	> 1%

12.3 Rate Limiting

```
# Test rate limiting by sending rapid requests
echo "=== Rate Limit Test (20 rapid requests) ==="
for i in $(seq 1 20); do
  CODE=$(curl -sk -X GET "${OFL_API}/version" \
    -H "Authorization: Bearer ${TOKEN}" \
    -o /dev/null -w "%{http_code}")
  echo "Request ${i}: HTTP ${CODE}"
done | sort | uniq -c | sort -rn
```

Expected output:

```
    20 Request: HTTP 200
```

If rate limiting is active, you may see `HTTP 429` (Too Many Requests) after a threshold.

Remediation: If the API is slow or unresponsive:
1. Check Apache and loginsight service health (Sections 4.1, 4.3)
2. Verify cluster health -- API calls are proxied to the master node
3. Check CPU and memory utilization on the master node
4. Review `/storage/var/loginsight/runtime.log` for API error messages
5. Restart Apache: `systemctl restart httpd`
6. As a last resort, restart the loginsight daemon: `systemctl restart loginsight`

13. Certificate Health

SSL/TLS certificates are critical for securing the Ops for Logs web UI, API, agent communication, and log forwarding. Expired or misconfigured certificates cause connection failures across the environment.

13.1 SSL Certificate Verification

Check the Web UI / API Certificate

```
# Inspect the SSL certificate served by Ops for Logs
echo | openssl s_client -connect ${OFL_HOST}:443 -servername ${OFL_HOST}
2>/dev/null | \
  openssl x509 -noout -subject -issuer -dates -serial -fingerprint -ext
subjectAltName
```

Expected output:

```
subject=CN = ops-for-logs.vcf.local
issuer=CN = VCF Internal CA, O = Virtual Control LLC, L = Managed
notBefore=Feb  1 00:00:00 2026 GMT
notAfter=Feb  1 23:59:59 2028 GMT
serial=4A3B2C1D0E9F8A7B
SHA256
Fingerprint=AB:CD:EF:12:34:56:78:9A:BC:DE:F0:12:34:56:78:9A:BC:DE:F0:12:34:56:78:9A:BC:
X509v3 Subject Alternative Name:
    DNS:ops-for-logs.vcf.local, DNS:ops-for-logs-node1.vcf.local, DNS:ops-for-logs-
node2.vcf.local, DNS:ops-for-logs-node3.vcf.local, IP Address:192.168.1.100, IP
Address:192.168.1.101, IP Address:192.168.1.102, IP Address:192.168.1.103
```

Check Certificate on Each Node

```
# Verify certificate consistency across all nodes
for NODE in ${OFL_NODE1} ${OFL_NODE2} ${OFL_NODE3}; do
  echo "===== ${NODE} ====="
  echo | openssl s_client -connect ${NODE}:443 -servername ${NODE} 2>/dev/null |
\
    openssl x509 -noout -subject -dates -fingerprint
  echo ""
done
```

Check Ingestion Port Certificate (9543)

```
# Verify the CFAPI ingestion port certificate
echo | openssl s_client -connect ${OFL_HOST}:9543 -servername ${OFL_HOST}
2>/dev/null | \
  openssl x509 -noout -subject -dates
```

13.2 Custom CA Configuration

```
# Check if a custom CA certificate is installed
ssh root@${OFL_NODE1} "
  echo '=== Custom CA Certificates ==='
  ls -la /storage/var/loginsight/certs/ 2>/dev/null
  echo ''
  echo '=== Trust Store Contents ==='
  keytool -list -keystore /storage/var/loginsight/certs/truststore.jks \
    -storepass changeit 2>/dev/null | head -20
"
```

Verify Full Certificate Chain

```
# Download and verify the full certificate chain
echo | openssl s_client -connect ${OFL_HOST}:443 -showcerts 2>/dev/null | \
  awk '/BEGIN CERTIFICATE/,/END CERTIFICATE/{ print }' > /tmp/ofl_chain.pem

# Verify the chain
openssl verify -verbose /tmp/ofl_chain.pem
```

13.3 Certificate Expiry Monitoring

Calculate Days Until Expiry

```
# Calculate days until certificate expiry
EXPIRY_DATE=$(echo | openssl s_client -connect ${OFL_HOST}:443 -servername
${OFL_HOST} 2>/dev/null | \
  openssl x509 -noout -enddate | cut -d= -f2)
EXPIRY_EPOCH=$(date -d "${EXPIRY_DATE}" +%s)
NOW_EPOCH=$(date +%s)
DAYS_REMAINING=$(( (EXPIRY_EPOCH - NOW_EPOCH) / 86400 ))
echo "Certificate expires: ${EXPIRY_DATE}"
echo "Days remaining: ${DAYS_REMAINING}"
```

Expected output:

```
Certificate expires: Feb  1 23:59:59 2028 GMT
Days remaining: 677
```

Check All Ports for Expiry

```
# Check certificate expiry on all service ports
echo "=== Certificate Expiry by Port ==="
for PORT in 443 9000 9543; do
  EXPIRY=$(echo | openssl s_client -connect ${OFL_HOST}:${PORT} 2>/dev/null | \
    openssl x509 -noout -enddate 2>/dev/null | cut -d= -f2)
  printf "Port %-6s Expires: %s\n" "${PORT}" "${EXPIRY:-N/A}"
done
```

Criteria	PASS	WARN	FAIL
Days until expiry	> 30 days	7-30 days	< 7 days or expired
SAN entries	Include VIP + all nodes	Missing some entries	Missing VIP or critical node
Certificate chain	Full chain valid	Intermediate missing (works)	Chain broken / untrusted
Consistency across nodes	Same cert on all nodes	--	Different certs on nodes

Ingestion port cert	Valid	Nearing expiry	Expired

Remediation: If certificates are expiring or invalid:
1. Generate a new CSR from the Ops for Logs admin UI: **Administration > SSL**
2. Submit the CSR to your CA and obtain the signed certificate
3. Upload the new certificate via the UI or API: `PUT /api/v1/ssl`
4. For custom CA trust, upload the CA certificate: `POST /api/v1/ssl/ca`
5. Restart Apache after certificate replacement: `systemctl restart httpd`
6. Verify all agents reconnect after certificate change -- agents must trust the new CA

14. NTP & DNS

Accurate time synchronization and reliable DNS resolution are foundational requirements for Ops for Logs. Time skew causes log correlation issues, and DNS failures prevent cluster communication.

14.1 Time Synchronization

Check NTP Status (chrony)

```
# Check chrony synchronization status on each node
ssh root@${OFL_NODE1} "chronyc tracking"
```

Expected output:

```
Reference ID    : C0A80001 (ntp-server.vcf.local)
Stratum         : 3
Ref time (UTC)  : Wed Mar 26 09:30:22 2026
System time     : 0.000023455 seconds fast of NTP time
Last offset     : +0.000012332 seconds
RMS offset      : 0.000034521 seconds
Frequency       : 2.345 ppm slow
Residual freq   : +0.001 ppm
Skew            : 0.023 ppm
Root delay      : 0.001234 seconds
Root dispersion : 0.000456 seconds
Update interval : 1024.0 seconds
Leap status     : Normal
```

Check NTP Sources

```
# List NTP sources and their status
ssh root@${OFL_NODE1} "chronyc sources -v"
```

Expected output:

```
  .-- Source mode  '^' = server, '=' = peer, '#' = local clock.
 / .- Source state '*' = current best, '+' = combined, '-' = not combined,
| /             'x' = may be in error, '~' = too variable, '?' = unusable.
||                                                 .- xxxx [ yyyy ] +/- zzzz
||      Reachability register (octal) -.           |  xxxx = adjusted offset,
||      Log2(Polling interval) --.      |          |  yyyy = measured offset,
||                                \     |          |  zzzz = estimated error.
```

```
||                                 |    |          \
MS Name/IP address         Stratum Poll Reach LastRx Last sample
===============================================================================
^* ntp-server.vcf.local          2  10   377   234   +0.012ms[ +0.015ms] +/-  1.23ms
^+ ntp-backup.vcf.local          2  10   377   512   -0.034ms[ -0.031ms] +/-  2.45ms
```

Compare Time Across All Nodes

```
# Check time offset between all nodes
echo "=== Time on each node ==="
for NODE in ${OFL_NODE1} ${OFL_NODE2} ${OFL_NODE3}; do
  TIME=$(ssh root@${NODE} "date -u '+%Y-%m-%d %H:%M:%S.%N UTC'")
  echo "${NODE}: ${TIME}"
done
```

Criteria	PASS	WARN	FAIL
NTP offset	< 100ms	100ms - 500ms	> 500ms
NTP source reachable	At least 1 source with `*`	Sources showing `?`	No reachable source
Inter-node time drift	< 200ms between nodes	200ms - 1s	> 1s between nodes
Leap status	`Normal`	--	`Not synchronised`

Remediation: If NTP is out of sync:
1. Force an immediate sync: `chronyc makestep`
2. Verify NTP server is reachable: `ping ntp-server.vcf.local`
3. Check chrony configuration: `cat /etc/chrony.conf`
4. Restart chrony: `systemctl restart chronyd`
5. If using ntpd instead: `systemctl restart ntpd && ntpq -p`

14.2 DNS Resolution

Forward DNS Lookup

```
# Verify DNS resolution for all Ops for Logs FQDNs
echo "=== Forward DNS Lookups ==="
for FQDN in ${OFL_HOST} ${OFL_NODE1} ${OFL_NODE2} ${OFL_NODE3}; do
  IP=$(dig +short ${FQDN} 2>/dev/null)
  printf "%-45s -> %s\n" "${FQDN}" "${IP:-FAILED}"
done
```

Expected output:

```
=== Forward DNS Lookups ===
ops-for-logs.vcf.local                        -> 192.168.1.100
ops-for-logs-node1.vcf.local                  -> 192.168.1.101
ops-for-logs-node2.vcf.local                  -> 192.168.1.102
ops-for-logs-node3.vcf.local                  -> 192.168.1.103
```

Reverse DNS Lookup

```
# Verify reverse DNS for all node IPs
echo "=== Reverse DNS Lookups ==="
for IP in 192.168.1.100 192.168.1.101 192.168.1.102 192.168.1.103; do
  HOSTNAME=$(dig +short -x ${IP} 2>/dev/null)
  printf "%-18s -> %s\n" "${IP}" "${HOSTNAME:-FAILED}"
done
```

DNS Response Time

```
# Measure DNS resolution time
echo "=== DNS Response Time ==="
for FQDN in ${OFL_HOST} ${OFL_NODE1}; do
  TIME=$(dig ${FQDN} | grep "Query time" | awk '{print $4, $5}')
```

```
  printf "%-45s %s\n" "${FQDN}" "${TIME}"
done
```

DNS Configuration on Nodes

```
# Check DNS configuration on each node
ssh root@${OFL_NODE1} "cat /etc/resolv.conf"
```

Expected output:

```
search vcf.local
nameserver 192.168.1.10
nameserver 192.168.1.11
```

Criteria	PASS	WARN	FAIL
Forward DNS	All FQDNs resolve	Slow resolution (> 1s)	Any FQDN fails to resolve
Reverse DNS	All IPs resolve to correct FQDN	Missing reverse for VIP	Missing reverse for node
DNS response time	< 100ms	100ms - 1s	> 1s
DNS servers configured	2+ nameservers	1 nameserver	0 nameservers

Remediation: If DNS is failing:
1. Verify DNS server reachability: `ping 192.168.1.10`
2. Check `/etc/resolv.conf` for correct nameserver entries
3. Test with a specific DNS server: `dig @192.168.1.10 ops-for-logs.vcf.local`
4. Add missing DNS records (forward and reverse) in your DNS infrastructure
5. Clear DNS cache if applicable: `systemd-resolve --flush-caches`

15. Backup Configuration

Regular backups of Ops for Logs configuration and data are essential for disaster recovery. This section verifies backup configuration and recency.

15.1 Backup Status

Check Backup Configuration via CLI

```
# Check backup schedule and recent backup status
ssh root@${OFL_NODE1} "
  echo '=== Backup Configuration ==='
  grep -A 20 'backup' /storage/var/loginsight/config/loginsight-config.xml
2>/dev/null | head -25
  echo ''
  echo '=== Recent Backup Files ==='
  ls -lhrt /storage/var/loginsight/backups/ 2>/dev/null | tail -10
"
```

Check Backup via API

```
# Get backup configuration and status
curl -sk -X GET "${OFL_API}/backup" \
  -H "Authorization: Bearer ${TOKEN}" | jq '.'
```

Expected response:

```
{
  "enabled": true,
  "schedule": "DAILY",
  "lastBackupTime": "2026-03-25T02:00:00Z",
  "lastBackupStatus": "SUCCESS",
  "lastBackupSizeBytes": 245678901,
  "backupDestination": "/storage/var/loginsight/backups",
  "retentionCount": 7
}
```

15.2 Backup Location & Retention

```
# Verify backup destination is accessible and has space
ssh root@${OFL_NODE1} "
  echo '=== Backup Directory ==='
  ls -lh /storage/var/loginsight/backups/ 2>/dev/null
  echo ''
  echo '=== Total Backup Size ==='
  du -sh /storage/var/loginsight/backups/ 2>/dev/null
  echo ''
  echo '=== Backup Count ==='
  ls -1 /storage/var/loginsight/backups/*.tar.gz 2>/dev/null | wc -l
"
```

Expected output:

```
=== Backup Directory ===
-rw-r--r-- 1 root root 234M Mar 25 02:01 backup-2026-03-25.tar.gz
-rw-r--r-- 1 root root 231M Mar 24 02:01 backup-2026-03-24.tar.gz
-rw-r--r-- 1 root root 228M Mar 23 02:01 backup-2026-03-23.tar.gz

=== Total Backup Size ===
1.6G    /storage/var/loginsight/backups/

=== Backup Count ===
7
```

Criteria	PASS	WARN	FAIL
Backup configured	Enabled with schedule	--	Not configured
Last backup status	`SUCCESS`	--	`FAILED`
Last backup age	< 24 hours	1-7 days	> 7 days
Backup retention	>= 3 copies	1-2 copies	0 copies
Backup destination space	> 20% free	10-20% free	< 10% free

Remediation: If backups are not configured or failing:
1. Enable backups via the admin UI: **Administration > Configuration > Backup**

2. Configure via API: `PUT /api/v1/backup` with schedule and destination
3. For external backup, configure NFS mount for backup destination
4. If backups are failing, check disk space at the destination
5. Trigger a manual backup: `POST /api/v1/backup/trigger`
6. Verify backup integrity by testing a restore in a non-production environment

16. Resource Utilization

Monitoring CPU, memory, disk I/O, and JVM heap usage per node ensures Ops for Logs has adequate resources and is not approaching capacity limits.

16.1 CPU & Memory per Node

CPU Utilization

```
# Check CPU utilization on each node
for NODE in ${OFL_NODE1} ${OFL_NODE2} ${OFL_NODE3}; do
  echo "===== ${NODE} ====="
  ssh root@${NODE} "
    echo '--- CPU Summary (mpstat) ---'
    mpstat 1 3 | tail -1
    echo ''
    echo '--- Load Average ---'
    uptime
    echo ''
    echo '--- Top CPU Processes ---'
    ps aux --sort=-%cpu | head -6
  "
  echo ""
done
```

Expected output (per node):

```
===== ops-for-logs-node1.vcf.local =====
--- CPU Summary (mpstat) ---
Average:     all    22.15    0.00    3.45    0.12    0.00    0.00    0.00    0.00
74.28

--- Load Average ---
 09:45:12 up 45 days,  3:12,  1 user,  load average: 1.42, 1.38, 1.35

--- Top CPU Processes ---
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root      1842 18.2 52.3 8234560 4312340 ?   Sl   Mar20 3214:23
/usr/lib/loginsight/application/sbin/loginsight
root      1523 12.5 35.2 5234560 2903450 ?   Sl   Mar20 1823:45 /usr/bin/java -
Xms2048m -Xmx2048m (cassandra)
root      2103  2.1  4.3  234560  354340 ?   Ss   Mar20  302:12 /usr/sbin/httpd
root      1955  1.3  1.6  198450  132340 ?   Sl   Mar20  189:34 /usr/bin/ruby
(fluentd)
```

Memory Utilization

```
# Check memory utilization on each node
for NODE in ${OFL_NODE1} ${OFL_NODE2} ${OFL_NODE3}; do
  echo "===== ${NODE} ====="
  ssh root@${NODE} "free -m"
  echo ""
done
```

Expected output (per node):

```
===== ops-for-logs-node1.vcf.local =====
               total        used        free      shared  buff/cache   available
Mem:           16016       10452        1234         128        4330        5184
Swap:           2048           0        2048
```

Criteria	PASS	WARN	FAIL
CPU utilization	< 70% sustained	70-90% sustained	> 90% sustained
Load average	< (CPU count * 0.7)	< (CPU count * 1.0)	> (CPU count * 1.5)
Memory used	< 80% of total	80-90% of total	> 90% of total
Swap usage	0 MB	< 500 MB	> 500 MB (indicates memory pressure)

16.2 JVM Heap Usage

Cassandra runs on the JVM and is sensitive to heap exhaustion. Log Insight also uses Java components.

Cassandra JVM Heap

```
# Check Cassandra JVM heap usage via nodetool
ssh root@${OFL_NODE1} "nodetool info | grep -E 'Heap|Off'"
```

Expected output:

```
Heap Memory (MB)    : 2048.00 / 2048.00
Off Heap Memory (MB): 123.45
```

Check for JVM Garbage Collection Issues

```
# Check Cassandra GC log for long pauses
ssh root@${OFL_NODE1} "grep -c 'GC pause.*[0-9]\{4,\}ms'
/storage/var/cassandra/logs/gc.log 2>/dev/null || echo '0'"

# Check for OutOfMemoryError
ssh root@${OFL_NODE1} "grep -c 'OutOfMemoryError'
/storage/var/cassandra/logs/system.log 2>/dev/null || echo '0'"
```

Log Insight JVM Heap

```
# Check Log Insight JVM heap from runtime log
ssh root@${OFL_NODE1} "grep -i 'heap\|memory' /storage/var/loginsight/runtime.log
| tail -10"
```

Criteria	PASS	WARN	FAIL
Cassandra heap usage	< 75% of max	75-90%	> 90% or OOM errors
GC pause duration	< 500ms	500ms - 2s	> 2s (application stalls)

GC pause frequency	< 1 per minute	1-5 per minute	> 5 per minute
OOM errors	0	--	Any OOM errors

16.3 Disk I/O Performance

```
# Check disk I/O statistics
ssh root@${OFL_NODE1} "
  echo '=== Disk I/O Stats (iostat) ==='
  iostat -xz 1 3 | tail -10
  echo ''
  echo '=== Disk Queue Depth ==='
  iostat -x | grep -E 'sdb|nvme' | awk '{print \$1, \"await:\" \$10 \"ms\",
\"util:\" \$NF \"%\"}'
"
```

Expected output:

```
=== Disk I/O Stats (iostat) ===
Device            r/s     w/s   rkB/s     wkB/s  await  %util
sdb             45.23   128.67  2345.00  8765.00   2.34  18.56

=== Disk Queue Depth ===
sdb await: 2.34ms util: 18.56%
```

Criteria	PASS	WARN	FAIL
Disk utilization (`%util`)	< 60%	60-85%	> 85%
Average wait (`await`)	< 10ms	10-50ms	> 50ms
I/O queue depth	< 4	4-16	> 16

Remediation: If resource utilization is high:
1. **CPU:** Identify top processes. If Cassandra, check compaction. If loginsight, check ingestion rate.
2. **Memory:** Increase VM memory allocation and adjust JVM heap (-Xmx) accordingly.
3. **Swap:** Any swap usage indicates memory pressure -- increase RAM.
4. **Disk I/O:** Migrate to faster storage (SSD/NVMe). Reduce retention period. Enable compression.
5. **JVM Heap:** Increase Cassandra heap in `/storage/var/cassandra/conf/cassandra-env.sh`. Restart Cassandra after changes.

17. Port Reference Table

The following table documents all network ports used by VCF Operations for Logs. Ensure firewall rules permit these ports between the listed source and destination components.

Port	Protocol	Direction	Source	Destination	Purpose
443	TCP (HTTPS)	Inbound	Browsers, API clients	Ops for Logs VIP/Nodes	Web UI and REST API access
80	TCP (HTTP)	Inbound	Browsers	Ops for Logs VIP/Nodes	HTTP redirect to HTTPS
9000	TCP	Inbound	Ops for Logs agents	Ops for Logs VIP/Nodes	CFAPI log ingestion (non-TLS)
9543	TCP (TLS)	Inbound	Ops for Logs agents	Ops for Logs VIP/Nodes	CFAPI log ingestion (TLS)
514	TCP/UDP	Inbound	Syslog sources	Ops for Logs VIP/Nodes	Syslog ingestion (non-TLS)
1514	TCP	Inbound	Syslog sources	Ops for Logs VIP/Nodes	Syslog ingestion (alternate port)
6514	TCP (TLS)	Inbound	Syslog sources	Ops for Logs VIP/Nodes	Syslog ingestion (TLS)
7000	TCP	Inter-node	Ops for Logs Node	Ops for Logs Node	Cassandra inter-node gossip
7001	TCP (TLS)	Inter-node	Ops for Logs Node	Ops for Logs Node	Cassandra inter-node TLS gossip
7199	TCP	Inter-node	Ops for Logs Node	Ops for Logs Node	Cassandra JMX monitoring
9042	TCP	Inter-node	Ops for Logs Node	Ops for Logs Node	Cassandra CQL native transport
9160	TCP	Inter-node	Ops for Logs Node	Ops for Logs Node	Cassandra Thrift client (legacy)
16520	TCP	Inter-node	Ops for Logs Node	Ops for Logs Node	Cluster replication and sync
16521	TCP (TLS)	Inter-node	Ops for Logs Node	Ops for Logs Node	Cluster replication (TLS)
123	UDP	Outbound	Ops for Logs Nodes	NTP Server	Time synchronization

53	TCP/UDP	Outbound	Ops for Logs Nodes	DNS Server	DNS resolution
389	TCP	Outbound	Ops for Logs Nodes	LDAP/AD Server	LDAP authentication
636	TCP (TLS)	Outbound	Ops for Logs Nodes	LDAP/AD Server	LDAPS authentication
25	TCP	Outbound	Ops for Logs Nodes	SMTP Server	Email notifications/alerts
587	TCP (TLS)	Outbound	Ops for Logs Nodes	SMTP Server	Email (TLS STARTTLS)
514/6514	TCP	Outbound	Ops for Logs Nodes	Forwarding destination	Log forwarding (syslog)
9543	TCP (TLS)	Outbound	Ops for Logs Nodes	Forwarding destination	Log forwarding (CFAPI)
443	TCP (HTTPS)	Outbound	Ops for Logs Nodes	VCF Operations	Integration with Ops Manager
443	TCP (HTTPS)	Outbound	Ops for Logs Nodes	vCenter Server	vSphere integration
443	TCP (HTTPS)	Outbound	Ops for Logs Nodes	SDDC Manager	VCF lifecycle management
443	TCP (HTTPS)	Outbound	Ops for Logs Nodes	Workspace ONE Access	VIDM SSO authentication
2049	TCP	Outbound	Ops for Logs Nodes	NFS Server	Archive storage (NFS)

Port Verification Script

```
# Verify all critical ports are listening on a node
ssh root@${OFL_NODE1} "
  echo '=== Listening Ports ==='
  ss -tuln | grep -E ':(443|80|9000|9543|514|1514|6514|7000|7199|9042|16520) ' |
sort -t: -k2 -n
"
```

Expected output:

```
tcp   LISTEN  0  128  *:80     *:*
tcp   LISTEN  0  128  *:443    *:*
tcp   LISTEN  0  128  *:514    *:*
tcp   LISTEN  0  128  *:1514   *:*
tcp   LISTEN  0  128  *:6514   *:*
tcp   LISTEN  0  128  *:7000   *:*
tcp   LISTEN  0  128  *:7199   *:*
tcp   LISTEN  0  128  *:9000   *:*
tcp   LISTEN  0  128  *:9042   *:*
tcp   LISTEN  0  128  *:9543   *:*
tcp   LISTEN  0  128  *:16520  *:*
```

Firewall Rule Validation

```
# Check iptables rules (if applicable)
ssh root@${OFL_NODE1} "iptables -L -n --line-numbers 2>/dev/null | head -40 ||
echo 'iptables not active'"

# Test external connectivity to key ports
for PORT in 443 9000 9543 514 6514; do
  nc -zv ${OFL_HOST} ${PORT} 2>&1 | grep -E 'succeeded|refused|timed'
done
```

18. Common Issues & Remediation

This section provides detailed troubleshooting guidance for the most frequently encountered Ops for Logs problems.

18.1 Cassandra Issues

18.1.1 Cassandra Fails to Start

Symptoms: `systemctl status cassandra` shows `failed`. Log queries return errors. Web UI shows "Service Unavailable".

Diagnosis:

```
# Check Cassandra system log for startup errors
ssh root@${OFL_NODE1} "tail -100 /storage/var/cassandra/logs/system.log | grep -i 'error\|exception\|fatal'"

# Check for commit log corruption
ssh root@${OFL_NODE1} "ls -la /storage/var/cassandra/commitlog/"

# Check disk space
ssh root@${OFL_NODE1} "df -h /storage/var"
```

Remediation:
1. If disk is full, free space by reducing retention or removing old archives
2. If commit log is corrupt, move corrupt files (do NOT delete): `mkdir /tmp/corrupt-cl && mv /storage/var/cassandra/commitlog/CommitLog-*.log /tmp/corrupt-cl/`
3. If JVM heap is insufficient, increase in `/storage/var/cassandra/conf/cassandra-env.sh`
4. Restart Cassandra: `systemctl restart cassandra`
5. Verify ring status: `nodetool status` -- ensure all nodes rejoin

18.1.2 Cassandra High Compaction Backlog

Symptoms: Slow queries, high disk I/O, increasing disk usage despite stable ingestion.

```
# Check compaction backlog
ssh root@${OFL_NODE1} "nodetool compactionstats"

# Check compaction throughput
ssh root@${OFL_NODE1} "nodetool getcompactionthroughput"
```

Remediation:
1. Temporarily increase compaction throughput: `nodetool setcompactionthroughput 256` (default is 64 MB/s)
2. Do NOT restart Cassandra during active compactions
3. Monitor progress: `watch -n 10 'nodetool compactionstats'`
4. If compaction is stuck, identify and remove stale SSTables (advanced, contact support)

18.1.3 Cassandra Node Shows DN (Down Normal)

Symptoms: `nodetool status` shows a node as `DN`. Cluster is degraded.

```
# Check connectivity to the down node
ping ${OFL_NODE2}
nc -zv ${OFL_NODE2} 7000
nc -zv ${OFL_NODE2} 9042

# Check logs on the down node
ssh root@${OFL_NODE2} "systemctl status cassandra && tail -50
/storage/var/cassandra/logs/system.log"
```

Remediation:
1. Verify network connectivity between nodes
2. Restart Cassandra on the down node: `systemctl restart cassandra`
3. Monitor it rejoining the ring: `nodetool status` (wait for `UJ` then `UN`)
4. If the node cannot rejoin, check for clock skew (Section 14.1)
5. As a last resort, decommission and recommission the node

18.2 Ingestion Drops

Symptoms: Missing logs in queries, ingestion EPS drops to zero or significantly below baseline, monitoring alerts on dropped events.

Diagnosis:

```
# Check for ingestion errors in runtime log
ssh root@${OFL_NODE1} "grep -i 'drop\|overflow\|backpressure\|reject' \
  /storage/var/loginsight/runtime.log | tail -20"

# Check ingestion pipeline ports
ssh root@${OFL_NODE1} "ss -tuln | grep -E ':(514|1514|6514|9000|9543)'"

# Check stats API for dropped events
curl -sk -X GET "${OFL_API}/stats" \
  -H "Authorization: Bearer ${TOKEN}" | jq '{droppedEvents,
currentEventsPerSecond, queueDepth}'
```

Remediation:
1. **Disk full:** The most common cause. Free disk space immediately (Section 6).
2. **Cassandra down:** Ops for Logs cannot index events if Cassandra is unhealthy (Section 18.1).
3. **Network saturation:** Check bandwidth utilization on ingestion NICs.
4. **Too many sources:** Add worker nodes to distribute ingestion load.
5. **Firewall blocking:** Verify ingestion ports are open from all log sources.
6. **Agent misconfiguration:** Verify agent destination points to VIP, not individual node.

18.3 Disk Full Scenarios

Symptoms: Ingestion halts, web UI errors, Cassandra write failures, `df -h /storage/var` shows > 95%.

Emergency Diagnosis:

```
# Identify what is consuming space
ssh root@${OFL_NODE1} "
  df -h /storage/var
  echo ''
  du -sh /storage/var/*/ 2>/dev/null | sort -rh
  echo ''
  echo '=== Largest files ==='
  find /storage/var -type f -size +1G -exec ls -lh {} \; 2>/dev/null | sort -k5 -rh | head -10
"
```

CRITICAL: Disk full is an emergency situation. Ops for Logs will stop ingesting logs and may become unresponsive. Address immediately.

Emergency Remediation (in priority order):

1. **Clear Cassandra snapshots:** `nodetool clearsnapshot` -- can free significant space
2. **Reduce retention period:** Temporarily reduce to force purge of old data
3. **Clear old archives:** If archiving to local disk, remove old archive files
4. **Remove core dumps:** `find /storage/var -name "core.*" -delete`
5. **Clear Fluentd buffers:** `rm -f /storage/var/fluentd/buffer/*.log`
6. **Expand the disk:** In vSphere, increase the VMDK size, then:
`growpart /dev/sdb 1 && resize2fs /dev/sdb1`
7. **Add NFS archive:** Move old data offload to NFS to free local space

18.4 Cluster Split-Brain

Symptoms: Two nodes claim to be master, data inconsistency between nodes, cluster API shows conflicting information.

Diagnosis:

```
# Check cluster state from each node
for NODE in ${OFL_NODE1} ${OFL_NODE2} ${OFL_NODE3}; do
  echo "===== ${NODE} ====="
  ssh root@${NODE} "curl -sk https://localhost/api/v1/cluster 2>/dev/null |
python3 -m json.tool | grep -E 'role|status'"
  echo ""
done

# Check Cassandra ring consistency
for NODE in ${OFL_NODE1} ${OFL_NODE2} ${OFL_NODE3}; do
  echo "===== ${NODE} ====="
  ssh root@${NODE} "nodetool describecluster | head -10"
  echo ""
done
```

CRITICAL: Split-brain is a serious condition that can cause data loss. Do NOT attempt to resolve without understanding which node has the most recent valid data.

Remediation:
1. **Identify the legitimate master:** The node with the most recent successful writes is typically authoritative
2. **Stop the false master:** `systemctl stop loginsight` on the node incorrectly claiming master
3. **Verify Cassandra consistency:** `nodetool repair` on the remaining nodes
4. **Restart the stopped node as worker:** The node should rejoin as a worker
5. **Check NTP:** Clock skew is a common cause of split-brain
6. **Check network partitions:** Ensure all nodes can reach each other on all required ports
7. **Contact VMware Support** if the cluster cannot self-heal

18.5 Certificate Problems

Symptoms: Browser SSL warnings, agent connection failures, API calls return TLS errors, forwarding breaks.

Diagnosis:

```
# Check certificate details
echo | openssl s_client -connect ${OFL_HOST}:443 2>&1 | grep -E
'Verify|depth|error|subject'

# Check certificate expiry
echo | openssl s_client -connect ${OFL_HOST}:443 2>/dev/null | openssl x509 -
noout -dates

# Check if agents can connect (from an agent host)
openssl s_client -connect ${OFL_HOST}:9543 </dev/null 2>&1 | grep "Verify return
code"
```

Remediation:

1. **Expired certificate:** Replace immediately via **Administration > SSL** in the UI
2. **Untrusted CA:** Upload the CA certificate to the trust store: `POST /api/v1/ssl/ca`
3. **SAN mismatch:** Regenerate the certificate with correct Subject Alternative Names
4. **Agent trust:** Deploy the CA certificate to all agent hosts. For ESXi: upload to `/etc/vmware/ssl/`
5. **After certificate change:** Restart Apache (`systemctl restart httpd`) and verify agents reconnect

18.6 Agent Disconnects

Symptoms: Agents showing `DISCONNECTED` status, gaps in log data from specific hosts, agent heartbeat timeouts.

Diagnosis (from the agent host):

```
# Check agent status on the remote host
ssh root@<agent-host> "systemctl status liagentd"

# Check agent log
ssh root@<agent-host> "tail -50 /var/log/liagent/liagent.log"

# Test connectivity to Ops for Logs
ssh root@<agent-host> "nc -zv ${OFL_HOST} 9543 && nc -zv ${OFL_HOST} 443"

# Check agent configuration
ssh root@<agent-host> "cat /var/lib/liagent/liagent.ini | grep -v '^;' | grep -v
'^$'"
```

On ESXi hosts:

```
# Check ESXi syslog configuration
ssh root@<esxi-host> "esxcli system syslog config get"

# Check ESXi Log Insight agent
ssh root@<esxi-host> "esxcli software vib list | grep -i loginsight"

# Test connectivity from ESXi
ssh root@<esxi-host> "nc -zv ${OFL_HOST} 9543"
```

Remediation:
1. **Agent not running:** Restart: `systemctl restart liagentd`
2. **Connectivity blocked:** Check firewall rules between agent and Ops for Logs (port 9543)
3. **Certificate trust:** Ensure the agent trusts the Ops for Logs CA
4. **Wrong destination:** Update `liagent.ini` to point to the VIP: `hostname=ops-for-logs.vcf.local`
5. **ESXi agent outdated:** Update the VIB: `esxcli software vib update -d /path/to/VMware-loginsight-agent.zip`
6. **DNS issue:** Verify the agent can resolve the Ops for Logs FQDN

19. CLI Quick Reference Card

This section provides a consolidated list of all CLI commands used throughout this handbook for quick reference.

System Service Commands

Command	Purpose
`systemctl status loginsight`	Check Log Insight daemon status
`systemctl status cassandra`	Check Cassandra service status
`systemctl status httpd`	Check Apache HTTPD status
`systemctl status fluentd`	Check Fluentd status
`systemctl restart loginsight`	Restart the Log Insight daemon
`systemctl restart cassandra`	Restart Cassandra
`systemctl restart httpd`	Restart Apache
`systemctl restart fluentd`	Restart Fluentd
`systemctl restart chronyd`	Restart NTP (chrony)
`journalctl -u loginsight --no-pager -n 100`	View recent Log Insight journal entries
`journalctl -u cassandra --no-pager -n 100`	View recent Cassandra journal entries

Cassandra (nodetool) Commands

Command	Purpose
`nodetool status`	Show Cassandra ring status and node states
`nodetool info`	Show node info including heap memory
`nodetool compactionstats`	Show pending and active compactions
`nodetool getcompactionthroughput`	Show current compaction throughput limit
`nodetool setcompactionthroughput <MB/s>`	Set compaction throughput (e.g., 128 or 256)
`nodetool describecluster`	Show cluster name, snitch, and schema versions
`nodetool repair`	Run a repair on the local node
`nodetool clearsnapshot`	Clear all saved snapshots to free disk space

`nodetool tpstats`	Show thread pool statistics
`nodetool cfstats`	Show column family (table) statistics
`nodetool gcstats`	Show garbage collection statistics

Storage & Disk Commands

Command	Purpose
`df -hT`	Show all filesystem usage with type
`df -h /storage/var`	Show `/storage/var` usage
`df -i /storage/var`	Show inode usage
`du -sh /storage/var/*/`	Show top-level directory sizes
`du -sh /storage/var/cassandra/data/`	Show Cassandra data size
`du -sh /storage/var/loginsight/`	Show Log Insight data size
`du -sh /storage/var/fluentd/buffer/`	Show Fluentd buffer size
`iostat -xz 1 3`	Show disk I/O statistics (3 samples)

Network & Connectivity Commands

Command	Purpose
`ss -tuln`	Show all listening TCP/UDP ports
`ss -tn`	Show all active TCP connections
`ss -s`	Show socket statistics summary
`nc -zv <host> <port>`	Test TCP connectivity to a specific port
`ping <host>`	Test ICMP reachability
`dig <fqdn>`	Forward DNS lookup
`dig +short <fqdn>`	Forward DNS lookup (short output)
`dig +short -x <ip>`	Reverse DNS lookup
`ip addr show`	Show network interface addresses
`arping -D -I eth0 <ip>`	Check for IP address conflicts

Certificate Commands

Command	Purpose
`openssl s_client -connect <host>:443`	Inspect the SSL certificate on port 443
`openssl x509 -noout -subject -dates -issuer`	Parse certificate details (piped from s_client)
`openssl x509 -noout -enddate`	Show only the expiry date
`openssl s_client -connect <host>:443 -showcerts`	Show the full certificate chain
`openssl verify <cert.pem>`	Verify a certificate chain
`keytool -list -keystore <path> -storepass changeit`	List Java trust store contents

Time Synchronization Commands

Command	Purpose
`chronyc tracking`	Show NTP tracking status
`chronyc sources -v`	Show NTP sources with details
`chronyc makestep`	Force an immediate time sync
`ntpq -p`	Show NTP peers (if using ntpd)
`date -u`	Show current UTC time
`timedatectl status`	Show time/date configuration

Process & Resource Commands

Command	Purpose
`ps aux --sort=-%cpu | head -10`	Top 10 processes by CPU
`ps aux --sort=-%mem | head -10`	Top 10 processes by memory
`free -m`	Show memory usage in MB
`uptime`	Show uptime and load average
`mpstat 1 3`	Show CPU statistics (3 samples)
`top -bn1 | head -20`	One-shot top output

Log File Locations

Log File	Purpose

`/storage/var/loginsight/runtime.log`	Main Ops for Logs application log
`/storage/var/cassandra/logs/system.log`	Cassandra system log
`/storage/var/cassandra/logs/gc.log`	Cassandra garbage collection log
`/var/log/httpd/error_log`	Apache error log
`/var/log/httpd/access_log`	Apache access log
`/var/log/fluentd/fluentd.log`	Fluentd log
`/var/log/liagent/liagent.log`	Log Insight agent log (on agent hosts)

20. API Quick Reference

All API endpoints use the base URL `https://<ops-for-logs-vip>/api/v1/`. Authentication is required for most endpoints via the `Authorization: Bearer <token>` header.

Authentication

```
# POST /api/v1/sessions -- Authenticate and obtain a session token
curl -sk -X POST "https://${OFL_HOST}/api/v1/sessions" \
  -H "Content-Type: application/json" \
  -d '{
    "username": "admin",
    "password": "<PASSWORD>",
    "provider": "Local"
  }'
# Response: { "sessionId": "<TOKEN>", "userId": "<UUID>", "ttl": 1800 }

# DELETE /api/v1/sessions/current -- Invalidate the current session
curl -sk -X DELETE "https://${OFL_HOST}/api/v1/sessions/current" \
  -H "Authorization: Bearer ${TOKEN}"
```

Version & System Info

```
# GET /api/v1/version -- Get product version info
curl -sk -X GET "https://${OFL_HOST}/api/v1/version" \
```

```
  -H "Authorization: Bearer ${TOKEN}" | jq '.'
# Response: { "version": "9.0.0", "build": "12345678", "releaseName": "..." }
```

Cluster Management

```
# GET /api/v1/cluster -- Get cluster configuration and node list
curl -sk -X GET "https://${OFL_HOST}/api/v1/cluster" \
  -H "Authorization: Bearer ${TOKEN}" | jq '.'

# GET /api/v1/cluster/status -- Get detailed cluster health status
curl -sk -X GET "https://${OFL_HOST}/api/v1/cluster/status" \
  -H "Authorization: Bearer ${TOKEN}" | jq '.'

# GET /api/v1/ilb -- Get ILB configuration
curl -sk -X GET "https://${OFL_HOST}/api/v1/ilb" \
  -H "Authorization: Bearer ${TOKEN}" | jq '.'
```

Statistics & Monitoring

```
# GET /api/v1/stats -- Get ingestion statistics
curl -sk -X GET "https://${OFL_HOST}/api/v1/stats" \
  -H "Authorization: Bearer ${TOKEN}" | jq '.'
# Response: { "totalEventsIngested": N, "currentEventsPerSecond": N,
"droppedEvents": N, ... }

# POST /api/v1/events/stats -- Query historical ingestion statistics
curl -sk -X POST "https://${OFL_HOST}/api/v1/events/stats" \
  -H "Authorization: Bearer ${TOKEN}" \
  -H "Content-Type: application/json" \
  -d '{
    "query": "ingestion_rate",
    "startTimeMillis": 1711411200000,
    "endTimeMillis": 1711497600000,
    "bucketDurationMinutes": 60
  }' | jq '.'
```

Event Queries

```
# POST /api/v1/events -- Search for events
curl -sk -X POST "https://${OFL_HOST}/api/v1/events" \
  -H "Authorization: Bearer ${TOKEN}" \
  -H "Content-Type: application/json" \
  -d '{
    "query": "vmw_vc_*",
    "startTimeMillis": 1711411200000,
    "endTimeMillis": 1711497600000,
    "limit": 100
  }' | jq '.'

# POST /api/v1/events/ingest/0 -- Ingest events via API
curl -sk -X POST "https://${OFL_HOST}/api/v1/events/ingest/0" \
  -H "Authorization: Bearer ${TOKEN}" \
  -H "Content-Type: application/json" \
  -d '{
    "events": [
      {
        "text": "Test event from API",
        "source": "api-test",
        "fields": [{"name": "env", "content": "production"}]
      }
    ]
  }'
```

Log Forwarding

```
# GET /api/v1/forwarding -- List all forwarding destinations
curl -sk -X GET "https://${OFL_HOST}/api/v1/forwarding" \
  -H "Authorization: Bearer ${TOKEN}" | jq '.'

# GET /api/v1/forwarding/stats -- Get forwarding statistics
curl -sk -X GET "https://${OFL_HOST}/api/v1/forwarding/stats" \
  -H "Authorization: Bearer ${TOKEN}" | jq '.'

# POST /api/v1/forwarding -- Create a new forwarding destination
curl -sk -X POST "https://${OFL_HOST}/api/v1/forwarding" \
  -H "Authorization: Bearer ${TOKEN}" \
  -H "Content-Type: application/json" \
  -d '{
    "name": "New-SIEM",
```

```
    "host": "siem.vcf.local",
    "port": 6514,
    "protocol": "SYSLOG",
    "transport": "TCP-TLS",
    "enabled": true,
    "filter": "*"
  }' | jq '.'
```

Content Packs

```
# GET /api/v1/content/contentpack/list -- List installed content packs
curl -sk -X GET "https://${OFL_HOST}/api/v1/content/contentpack/list" \
  -H "Authorization: Bearer ${TOKEN}" | jq '.'

# GET /api/v1/content/contentpack/marketplace -- Check marketplace for updates
curl -sk -X GET "https://${OFL_HOST}/api/v1/content/contentpack/marketplace" \
  -H "Authorization: Bearer ${TOKEN}" | jq '.'

# GET /api/v1/content/contentpack/autoupdate -- Check auto-update configuration
curl -sk -X GET "https://${OFL_HOST}/api/v1/content/contentpack/autoupdate" \
  -H "Authorization: Bearer ${TOKEN}" | jq '.'

# PUT /api/v1/content/contentpack/autoupdate -- Enable/disable auto-update
curl -sk -X PUT "https://${OFL_HOST}/api/v1/content/contentpack/autoupdate" \
  -H "Authorization: Bearer ${TOKEN}" \
  -H "Content-Type: application/json" \
  -d '{"autoUpdateEnabled": true, "checkIntervalHours": 24}' | jq '.'
```

Agent Management

```
# GET /api/v1/agent/stats -- Get agent summary statistics
curl -sk -X GET "https://${OFL_HOST}/api/v1/agent/stats" \
  -H "Authorization: Bearer ${TOKEN}" | jq '.'

# GET /api/v1/agent/agents -- List all agents
curl -sk -X GET "https://${OFL_HOST}/api/v1/agent/agents" \
  -H "Authorization: Bearer ${TOKEN}" | jq '.'

# GET /api/v1/agent/agents?status=DISCONNECTED -- List disconnected agents
```

```
curl -sk -X GET "https://${OFL_HOST}/api/v1/agent/agents?status=DISCONNECTED" \
  -H "Authorization: Bearer ${TOKEN}" | jq '.'

# GET /api/v1/agent/groups -- List all agent groups
curl -sk -X GET "https://${OFL_HOST}/api/v1/agent/groups" \
  -H "Authorization: Bearer ${TOKEN}" | jq '.'

# GET /api/v1/agent/groups/<groupId> -- Get specific agent group configuration
curl -sk -X GET "https://${OFL_HOST}/api/v1/agent/groups/group-001" \
  -H "Authorization: Bearer ${TOKEN}" | jq '.'

# DELETE /api/v1/agent/agents/<agentId> -- Remove a stale agent
curl -sk -X DELETE "https://${OFL_HOST}/api/v1/agent/agents/<agentId>" \
  -H "Authorization: Bearer ${TOKEN}"
```

Integration

```
# GET /api/v1/integration/vrops -- Check VCF Operations integration status
curl -sk -X GET "https://${OFL_HOST}/api/v1/integration/vrops" \
  -H "Authorization: Bearer ${TOKEN}" | jq '.'

# GET /api/v1/auth/providers -- List authentication providers
curl -sk -X GET "https://${OFL_HOST}/api/v1/auth/providers" \
  -H "Authorization: Bearer ${TOKEN}" | jq '.'
```

SSL / Certificates

```
# GET /api/v1/ssl -- Get current SSL certificate information
curl -sk -X GET "https://${OFL_HOST}/api/v1/ssl" \
  -H "Authorization: Bearer ${TOKEN}" | jq '.'

# POST /api/v1/ssl/ca -- Upload a custom CA certificate
curl -sk -X POST "https://${OFL_HOST}/api/v1/ssl/ca" \
  -H "Authorization: Bearer ${TOKEN}" \
  -H "Content-Type: application/json" \
  -d '{"certificate": "<PEM-encoded-CA-cert>"}' | jq '.'

# PUT /api/v1/ssl -- Replace the server certificate
curl -sk -X PUT "https://${OFL_HOST}/api/v1/ssl" \
```

```
  -H "Authorization: Bearer ${TOKEN}" \
  -H "Content-Type: application/json" \
  -d '{
    "certificate": "<PEM-encoded-cert>",
    "privateKey": "<PEM-encoded-key>",
    "certificateChain": "<PEM-encoded-chain>"
  }' | jq '.'
```

Backup & Restore

```
# GET /api/v1/backup -- Get backup configuration
curl -sk -X GET "https://${OFL_HOST}/api/v1/backup" \
  -H "Authorization: Bearer ${TOKEN}" | jq '.'

# POST /api/v1/backup/trigger -- Trigger an immediate backup
curl -sk -X POST "https://${OFL_HOST}/api/v1/backup/trigger" \
  -H "Authorization: Bearer ${TOKEN}" | jq '.'

# PUT /api/v1/backup -- Configure backup settings
curl -sk -X PUT "https://${OFL_HOST}/api/v1/backup" \
  -H "Authorization: Bearer ${TOKEN}" \
  -H "Content-Type: application/json" \
  -d '{
    "enabled": true,
    "schedule": "DAILY",
    "retentionCount": 7,
    "backupDestination": "/storage/var/loginsight/backups"
  }' | jq '.'
```

Retention & Archive

```
# GET /api/v1/time/config -- Get retention configuration
curl -sk -X GET "https://${OFL_HOST}/api/v1/time/config" \
  -H "Authorization: Bearer ${TOKEN}" | jq '.'

# PUT /api/v1/time/config -- Update retention settings
curl -sk -X PUT "https://${OFL_HOST}/api/v1/time/config" \
  -H "Authorization: Bearer ${TOKEN}" \
  -H "Content-Type: application/json" \
```

```
  -d '{"retentionPeriod": 30}' | jq '.'

# GET /api/v1/archive -- Get archive configuration
curl -sk -X GET "https://${OFL_HOST}/api/v1/archive" \
  -H "Authorization: Bearer ${TOKEN}" | jq '.'
```

VCF Operations for Logs Health Check Handbook

Version 1.0 -- March 2026

For questions, updates, or feedback regarding this handbook, contact the VCF operations team.