HEALTH CHECK HANDBOOK

VIRTUAL CONTROL

VMWARE CLOUD FOUNDATION SOLUTIONS

vCenter Server Health Check Handbook

Comprehensive vCenter Server health validation including services, database, SSO, certificates, inventory, and performance assessment.

VCENTER | VPXD | DATABASE | SSO | CERTIFICATES

VCF 9.0 VMware Cloud Foundation

PROPRIETARY & CONFIDENTIAL

vCenter Server Health Check Handbook

Comprehensive Health Verification for vCenter Server in VCF 9

Author: Virtual Control LLC **Date:** March 2026 **Version:** 1.0 **Classification:** Internal Use **Platform:** VMware Cloud Foundation 9.0 / vCenter Server 8.x

Table of Contents

1. Overview & Purpose

This handbook provides a **complete, step-by-step health check procedure** for VMware vCenter Server 8.x deployed within a VCF 9.0 environment. It is designed for VMware administrators who need to verify vCenter Server health during:

- **Routine maintenance windows** -- Weekly or monthly proactive checks
- **Pre/post-upgrade validation** -- Before and after vCenter patches or upgrades
- **Incident troubleshooting** -- When service, connectivity, or performance issues occur
- **Environment handover** -- Documenting health state for audits or transfers
- **VCF lifecycle operations** -- Before and after SDDC Manager orchestrated updates

What This Document Covers

Area	Components Checked
Core Services	VPXD, vmon services, service-control status
Appliance Health	VAMI REST API health endpoints for system, memory, storage, load, swap
Database	Embedded PostgreSQL health, VCDB size, vacuum status
High Availability	VCHA mode, active/passive/witness, failover readiness
Certificates	Machine SSL, STS, VECS stores, certificate expiry
Storage	Disk partitions, log storage, database storage, filesystem utilization
Performance	CPU, memory, swap, load average via API and CLI
Identity	SSO domain, identity sources, LDAP/AD binding, token validation
Plugins	Registered extensions, plugin health, stale cleanup
Lookup Service	Service registrations, STS health, PSC endpoints
Inventory	Datacenter/cluster/host/VM counts, consistency
Logging	Syslog forwarding, log rotation, log bundle generation
Time Sync	NTP configuration, drift check

Health Check Methodology

Each check in this handbook follows a consistent format:

1. **What to check** -- Description of the component and why it matters
2. **How to check** -- Exact CLI command or API call (copy-paste ready)
3. **Expected output** -- What a healthy result looks like

4. **Pass / Warn / Fail criteria** -- Clear thresholds with visual indicators
5. **Remediation** -- What to do if the check fails

Environment Variables: Throughout this document, replace the following placeholders with your actual values:
`$VC_FQDN` = vCenter Server FQDN (e.g., vcenter01.lab.local)
`$VC_USER` = administrator@vsphere.local
`$VC_PASS` = vCenter SSO administrator password
`$VC_TOKEN` = Session token obtained via authentication API

2. Prerequisites

Required Access

Access Type	Details
SSH Access	Root shell access to vCenter Appliance (enable via VAMI if disabled)
VAMI Access	https://$VC_FQDN:5480 -- root credentials
vSphere Client	https://$VC_FQDN/ui -- administrator@vsphere.local
REST API	https://$VC_FQDN/api -- session-based authentication
SDDC Manager	For VCF-specific lifecycle checks

Required Tools

Tool	Purpose
`curl`	REST API calls from jump host or local machine
`jq`	JSON parsing of API responses
`openssl`	Certificate inspection and expiry checks
SSH client	Shell access for CLI commands
Web browser	VAMI and vSphere Client access

Environment Setup

Set these variables before running commands:

```
# vCenter connection variables
export VC_FQDN="vcenter01.lab.local"
export VC_USER="administrator@vsphere.local"
export VC_PASS='YourPasswordHere'

# Obtain a session token
export VC_TOKEN=$(curl -sk -X POST \
  "https://${VC_FQDN}/api/session" \
  -u "${VC_USER}:${VC_PASS}" | tr -d '"')
```

```
# Verify the token was obtained
echo "Session Token: ${VC_TOKEN}"
```

Security Note: Never store passwords in shell history. Use `read -s VC_PASS` for interactive password entry. Destroy sessions when finished with: `curl -sk -X DELETE "https://${VC_FQDN}/api/session" -H "vmware-api-session-id: ${VC_TOKEN}"`

Enable SSH on vCenter Appliance

If SSH is not enabled, enable it via VAMI:

1. Navigate to `https://$VC_FQDN:5480`
2. Log in with root credentials
3. Go to **Access** > **SSH Login** > **Edit** > Enable
4. Alternatively via API:

```
curl -sk -X PUT \
  "https://${VC_FQDN}/api/appliance/access/ssh" \
  -H "vmware-api-session-id: ${VC_TOKEN}" \
  -H "Content-Type: application/json" \
  -d '{"enabled": true}'
```

3. Quick Reference -- All Checks Summary

#	Check	Command / Endpoint	Pass	Warn	Fail
4.1	VPXD Process	`service-control --status vmware-vpxd`	RUNNING	--	STOPPED
5.1	All vCenter Services	`service-control --status --all`	All STARTED	1-2 non-critical stopped	Critical stopped
6.1	System Health (VAMI)	`/api/appliance/health/system`	green	yellow	orange/red
6.2	Memory Health	`/api/appliance/health/mem`	green	yellow	orange/red
6.3	Storage Health	`/api/appliance/health/storage`	green	yellow	orange/red
6.4	Database Storage	`/api/appliance/health/database-storage`	green	yellow	orange/red
6.5	CPU Load	`/api/appliance/health/load`	green	yellow	orange/red
6.6	Swap Health	`/api/appliance/health/swap`	green	yellow	orange/red
6.7	Software Packages	`/api/appliance/health/softwarepackages`	green	yellow	red
7.1	PostgreSQL Status	`systemctl status vmware-vpostgres`	active	--	inactive
7.2	VCDB Size	SQL query	<50GB	50-80GB	>80GB
8.1	VCHA Mode	`/api/vcenter/vcha/cluster/mode`	ENABLED	--	DISABLED
9.1	Machine SSL Cert	`openssl` expiry check	>60 days	30-60 days	<30 days
9.3	STS Certificate	`/usr/lib/vmware-vmca` check	>60 days	30-60 days	<30 days
10.1	Disk Utilization	`df -h`	<70%	70-85%	>85%

11.1	CPU Utilization	API + `top`	<70%	70-85%	>85%
11.2	Memory Utilization	API + `free`	<80%	80-90%	>90%
12.1	SSO Domain	`sso-config.sh`	Healthy	--	Error
14.1	Lookup Service	`lstool.py`	Registered	--	Missing
17.1	NTP Sync	`/api/appliance/ntp`	Synced	Drift >1s	Not configured

4. VPXD Service Health

The **VPXD** (VMware VirtualCenter Server Daemon) is the core service of vCenter Server. It manages ESXi hosts, virtual machines, storage, and networking. If VPXD is down, the entire vCenter is non-functional.

4.1 VPXD Process Status

CLI Check (SSH to vCenter Appliance)

```
# Check VPXD service status
service-control --status vmware-vpxd
```

Expected Output (Healthy):

```
VMware vCenter Server:Status: RUNNING
```

Alternative: systemctl check

```
systemctl status vmware-vpxd
```

Expected Output (Healthy):

```
● vmware-vpxd.service - VMware vCenter Server
   Loaded: loaded (/usr/lib/systemd/system/vmware-vpxd.service; enabled)
   Active: active (running) since Mon 2026-03-23 10:15:22 UTC; 3 days ago
 Main PID: 5432 (vpxd)
    Tasks: 312
   Memory: 2.1G
   CGroup: /system.slice/vmware-vpxd.service
           └─5432 /usr/lib/vmware-vpxd/vpxd
```

API Check

```
curl -sk -X GET \
  "https://${VC_FQDN}/api/vcenter/services/vmware-vpxd" \
```

```
  -H "vmware-api-session-id: ${VC_TOKEN}" | jq .
```

Condition	Result	Badge
Status = RUNNING, Active (running)	Healthy	PASS
Status = STARTING (briefly during boot)	Transitional	WARN
Status = STOPPED or inactive (dead)	Critical failure	FAIL

Remediation (VPXD Stopped):
1. Attempt to start: `service-control --start vmware-vpxd`
2. If fails, check logs: `tail -200 /var/log/vmware/vpxd/vpxd.log`
3. Check for port conflicts: `netstat -tlnp | grep 443`
4. Verify database connectivity: `systemctl status vmware-vpostgres`
5. If persistent, restart all services: `service-control --stop --all && service-control --start --all`

4.2 VPXD Restart Procedures

Warning: Restarting VPXD will disconnect all vSphere Client sessions and temporarily interrupt management operations. VMs continue to run on ESXi hosts unaffected.

Graceful Restart

```
# Stop then start VPXD
service-control --stop vmware-vpxd
sleep 10
service-control --start vmware-vpxd

# Verify it came back
service-control --status vmware-vpxd
```

Full Service Stack Restart

```
# Nuclear option -- restarts all vCenter services
service-control --stop --all
sleep 15
service-control --start --all
```

```
# Verify all services
service-control --status --all
```

4.3 VPXD Log Analysis

Key Log Files

Log File	Purpose
`/var/log/vmware/vpxd/vpxd.log`	Main VPXD log -- service operations, errors
`/var/log/vmware/vpxd/vpxd-alert.log`	Critical alerts only
`/var/log/vmware/vpxd/vpxd-profiler.log`	Performance profiling data
`/var/log/vmware/vpxd/vpxd-svcs.log`	Service-level operations

Check for Recent Errors

```
# Last 50 error entries in VPXD log
grep -i "error\|fatal\|panic\|exception" /var/log/vmware/vpxd/vpxd.log | tail -50
```

```
# Check for crash indicators
grep -c "core dump\|segfault\|SIGABRT" /var/log/vmware/vpxd/vpxd.log
```

```
# Check VPXD alert log
cat /var/log/vmware/vpxd/vpxd-alert.log | tail -20
```

Expected Output (Healthy): No recent fatal errors, no crash indicators, alert log empty or minimal entries.

Condition	Badge
No errors or only informational entries	PASS
Warning-level entries present	WARN
Fatal/crash/exception entries in last 24h	FAIL

5. vCenter Services Status

5.1 vmon-cli Service Listing

The `vmon-cli` utility manages vCenter services at the vMon (VMware Service Lifecycle Manager) level.

List All Services

```
vmon-cli --list
```

Expected Output (Healthy):

```
analyticss        STARTED
applmgmt          STARTED
certificateauthority STARTED
certificatemanagement STARTED
cis-license       STARTED
content-library   STARTED
eam               STARTED
envoy             STARTED
hvc               STARTED
imagebuilder      STARTED
infraprofile      STARTED
lookupsvc         STARTED
netdumper         STARTED
observability     STARTED
perfcharts        STARTED
pschealth         STARTED
rbd               STARTED
rhttpproxy        STARTED
sca               STARTED
sps               STARTED
statsmonitor      STARTED
sts               STARTED
topologysvc       STARTED
trustmanagement   STARTED
updatemgr         STARTED
vapi-endpoint     STARTED
vcha              STARTED
vlcm              STARTED
```

```
vmcam              STARTED
vmonapi            STARTED
vmware-vpostgres   STARTED
vpxd               STARTED
vpxd-svcs          STARTED
vsan-health        STARTED
vsm                STARTED
vsphere-ui         STARTED
vstats             STARTED
vtsdb              STARTED
wcp                STARTED
```

Check Specific Service Status

```
# Check a single service
vmon-cli --status vpxd

# Check multiple services
for svc in vpxd lookupsvc sts vmware-vpostgres vsphere-ui; do
  echo "$svc: $(vmon-cli --status $svc)"
done
```

5.2 service-control Commands

Check All Services Status

```
service-control --status --all
```

Expected Output (Healthy):

```
VMware vCenter Server:Status: RUNNING
VMware vAPI Endpoint:Status: RUNNING
VMware Content Library:Status: RUNNING
VMware Certificate Authority:Status: RUNNING
VMware Identity Management Service:Status: RUNNING
VMware Lookup Service:Status: RUNNING
VMware Security Token Service:Status: RUNNING
VMware vSphere Client:Status: RUNNING
VMware vSphere Update Manager:Status: RUNNING
```

```
VMware PostgreSQL:Status: RUNNING
VMware HTTP Reverse Proxy:Status: RUNNING
VMware Envoy Service:Status: RUNNING
...
(all services RUNNING)
```

Start / Stop / Restart Individual Services

```
# Stop a specific service
service-control --stop vmware-updatemgr

# Start a specific service
service-control --start vmware-updatemgr

# Restart a specific service (stop + start)
service-control --stop vmware-vsphere-ui && service-control --start vmware-
vsphere-ui
```

5.3 Critical vs Non-Critical Services

Service	Criticality	Impact if Stopped
`vpxd`	**CRITICAL**	vCenter completely non-functional
`vmware-vpostgres`	**CRITICAL**	Database unavailable, all services fail
`vmware-sts`	**CRITICAL**	SSO authentication fails, no logins
`lookupsvc`	**CRITICAL**	Service discovery fails
`rhttpproxy`	**CRITICAL**	All HTTPS endpoints inaccessible
`envoy`	**CRITICAL**	Reverse proxy down, API unreachable
`vpxd-svcs`	HIGH	vCenter sub-services degraded
`vsphere-ui`	HIGH	vSphere Client (HTML5) unavailable
`vapi-endpoint`	HIGH	REST API unavailable
`vmware-sps`	MEDIUM	Storage profile service down
`content-library`	MEDIUM	Content library operations fail
`updatemgr`	MEDIUM	vSphere Lifecycle Manager offline
`vlcm`	MEDIUM	Lifecycle operations unavailable

`eam`	MEDIUM	ESX Agent Manager down
`perfcharts`	LOW	Performance charts unavailable
`imagebuilder`	LOW	Image building unavailable
`vstats`	LOW	vStats collection paused
`netdumper`	LOW	Network core dump receiver offline
`analytics`	LOW	CEIP analytics paused

Condition	Badge
All services RUNNING	PASS
1-2 LOW/MEDIUM services stopped	WARN
Any CRITICAL/HIGH service stopped	FAIL

Remediation (Services Stopped):
1. Start individual service: `service-control --start <service-name>`
2. If dependency failure, start all: `service-control --start --all`
3. Check service logs: `journalctl -u <service-name> --no-pager -n 100`
4. Last resort full restart: `reboot` (from appliance shell)

6. vCenter Appliance Health (VAMI)

The VAMI (vCenter Server Appliance Management Interface) REST API provides health status for all key appliance subsystems. These endpoints return standardized color-coded health states: `green`, `yellow`, `orange`, `red`, `gray`.

> **Health Color Key:**
> `green` = Healthy | `yellow` = Warning, degraded | `orange` = Degraded, action needed | `red` = Critical failure | `gray` = Unknown / not available

6.1 System Health

```
curl -sk -X GET \
  "https://${VC_FQDN}/api/appliance/health/system" \
  -H "vmware-api-session-id: ${VC_TOKEN}"
```

Expected Output:

```
"green"
```

Detailed System Health (with messages)

```
curl -sk -X GET \
  "https://${VC_FQDN}/api/appliance/health/system?messages=true" \
  -H "vmware-api-session-id: ${VC_TOKEN}" | jq .
```

Condition	Badge
`"green"`	PASS
`"yellow"`	WARN
`"orange"` or `"red"`	FAIL

6.2 Memory Health

```
curl -sk -X GET \
  "https://${VC_FQDN}/api/appliance/health/mem" \
  -H "vmware-api-session-id: ${VC_TOKEN}"
```

Expected Output:

```
"green"
```

Condition	Threshold	Badge
`"green"`	Memory utilization < 80%	PASS
`"yellow"`	Memory utilization 80-95%	WARN
`"orange"` / `"red"`	Memory utilization > 95% or OOM	FAIL

Remediation (Memory Warning/Critical):
1. Check top memory consumers: SSH to appliance, run `top -o %MEM`
2. Restart heavy services: `service-control --stop vmware-vpxd && service-control --start vmware-vpxd`
3. Check for memory leaks in VPXD: `grep -i "out of memory\|oom" /var/log/vmware/vpxd/vpxd.log`
4. If persistent, increase appliance VM memory allocation (requires shutdown)

6.3 Storage Health

```
curl -sk -X GET \
  "https://${VC_FQDN}/api/appliance/health/storage" \
  -H "vmware-api-session-id: ${VC_TOKEN}"
```

Expected Output:

```
"green"
```

Condition	Threshold	Badge
`"green"`	All partitions below warning threshold	PASS
`"yellow"`	One or more partitions 70-85% full	WARN
`"orange"` / `"red"`	Partitions > 85% full	FAIL

6.4 Database Storage Health

```
curl -sk -X GET \
  "https://${VC_FQDN}/api/appliance/health/database-storage" \
  -H "vmware-api-session-id: ${VC_TOKEN}"
```

Expected Output:

```
"green"
```

Condition	Threshold	Badge
`"green"`	DB storage utilization < 70%	PASS
`"yellow"`	DB storage utilization 70-85%	WARN
`"orange"` / `"red"`	DB storage utilization > 85%	FAIL

6.5 CPU Load Health

```
curl -sk -X GET \
  "https://${VC_FQDN}/api/appliance/health/load" \
  -H "vmware-api-session-id: ${VC_TOKEN}"
```

Expected Output:

```
"green"
```

Condition	Threshold	Badge
`"green"`	Load average within normal range	PASS
`"yellow"`	Load average elevated	WARN
`"orange"` / `"red"`	Load critically high	FAIL

6.6 Swap Health

```
curl -sk -X GET \
  "https://${VC_FQDN}/api/appliance/health/swap" \
  -H "vmware-api-session-id: ${VC_TOKEN}"
```

Expected Output:

```
"green"
```

Condition	Threshold	Badge
`"green"`	Swap usage minimal or zero	PASS
`"yellow"`	Swap usage moderate	WARN
`"orange"` / `"red"`	Swap usage critically high	FAIL

Remediation (Swap Critical):
1. Check swap usage: `free -m` and `swapon --show`
2. Identify swap-heavy processes: `for pid in /proc/[0-9]*; do awk '/VmSwap/{print FILENAME,$0}' $pid/status 2>/dev/null; done | sort -k3 -rn | head -20`
3. If persistent, increase VM memory and reduce swap pressure by restarting services

6.7 Software Packages Health

```
curl -sk -X GET \
  "https://${VC_FQDN}/api/appliance/health/softwarepackages" \
  -H "vmware-api-session-id: ${VC_TOKEN}"
```

Expected Output:

```
"green"
```

Condition	Badge
`"green"` -- All packages consistent	PASS
`"yellow"` -- Minor package inconsistencies	WARN
`"red"` -- Package corruption or missing packages	FAIL

Comprehensive VAMI Health Script

Run all health checks at once:

```
echo "=== vCenter Appliance Health Summary ==="
for endpoint in system mem storage database-storage load swap softwarepackages;
do
  result=$(curl -sk -X GET \
    "https://${VC_FQDN}/api/appliance/health/${endpoint}" \
    -H "vmware-api-session-id: ${VC_TOKEN}" | tr -d '"')
  printf "%-25s : %s\n" "$endpoint" "$result"
done
```

Expected Output (All Healthy):

```
=== vCenter Appliance Health Summary ===
system                    : green
mem                       : green
storage                   : green
database-storage          : green
load                      : green
swap                      : green
softwarepackages          : green
```

7. Database Health

vCenter Server 8.x uses an embedded PostgreSQL database (vPostgres) for all configuration and inventory data. Database health is foundational to vCenter operations.

7.1 PostgreSQL Embedded DB Status

Service Status

```
# Check vPostgres service
systemctl status vmware-vpostgres
```

Expected Output (Healthy):

```
● vmware-vpostgres.service - VMware Postgres
   Loaded: loaded (/usr/lib/systemd/system/vmware-vpostgres.service; enabled)
   Active: active (running) since Mon 2026-03-23 10:14:55 UTC; 3 days ago
 Main PID: 4821 (postgres)
    Tasks: 48
   Memory: 512.3M
   CGroup: /system.slice/vmware-vpostgres.service
           ├─4821 /opt/vmware/vpostgres/current/bin/postgres -D /storage/db/vpostgres
           ├─4910 postgres: checkpointer
           ├─4911 postgres: background writer
           └─ ...
```

Test Database Connectivity

```
# Connect to VCDB and run a basic query
/opt/vmware/vpostgres/current/bin/psql -U postgres -d VCDB -c "SELECT version();"
```

Expected Output:

```
                                                 version
-------------------------------------------------------------------------------------------------------
 PostgreSQL 14.x (VMware Postgres 14.x) on x86_64-unknown-linux-gnu, compiled by gcc
```

```
...
(1 row)
```

Condition	Badge
Service active (running), query succeeds	PASS
Service active but slow queries	WARN
Service inactive or query fails	FAIL

7.2 VCDB Size Monitoring

```
# Check total VCDB size
/opt/vmware/vpostgres/current/bin/psql -U postgres -d VCDB -c \
  "SELECT pg_size_pretty(pg_database_size('VCDB')) AS db_size;"
```

Expected Output:

```
 db_size
---------
 12 GB
(1 row)
```

Top Tables by Size

```
/opt/vmware/vpostgres/current/bin/psql -U postgres -d VCDB -c "
SELECT
  schemaname || '.' || tablename AS table_full_name,
  pg_size_pretty(pg_total_relation_size(schemaname || '.' || tablename)) AS
total_size
FROM pg_tables
WHERE schemaname NOT IN ('pg_catalog', 'information_schema')
ORDER BY pg_total_relation_size(schemaname || '.' || tablename) DESC
LIMIT 15;"
```

Expected Output:

```
         table_full_name          | total_size
----------------------------------+------------
```

```
 vc.vpx_event_arg                | 3200 MB
 vc.vpx_event                    | 2100 MB
 vc.vpx_task_event               | 1800 MB
 vc.vpx_stat_counter             | 980 MB
 vc.vpx_task                     | 850 MB
 ...
(15 rows)
```

Condition	Badge
VCDB size < 50 GB	PASS
VCDB size 50 - 80 GB	WARN
VCDB size > 80 GB	FAIL

Remediation (Database Too Large):
1. Purge old events and tasks via vSphere Client: **Administration > vCenter Server Settings > Runtime Settings**
2. Reduce task and event retention: set `task.maxAge` and `event.maxAge` to 30 days
3. Manual cleanup (careful!): `/opt/vmware/vpostgres/current/bin/psql -U postgres -d VCDB -c "DELETE FROM vc.vpx_event WHERE create_time < NOW() - INTERVAL '30 days';"`
4. Run vacuum afterward: `/opt/vmware/vpostgres/current/bin/psql -U postgres -d VCDB -c "VACUUM FULL ANALYZE;"`

7.3 Vacuum & Maintenance

Check Last Vacuum Time

```
/opt/vmware/vpostgres/current/bin/psql -U postgres -d VCDB -c "
SELECT
  schemaname || '.' || relname AS table_name,
  last_vacuum,
  last_autovacuum,
  last_analyze,
  n_dead_tup
FROM pg_stat_user_tables
WHERE n_dead_tup > 1000
ORDER BY n_dead_tup DESC
LIMIT 10;"
```

Expected Output:

```
      table_name        |      last_vacuum      |   last_autovacuum    |
last_analyze      | n_dead_tup
------------------------+-----------------------+----------------------+------------
----------+------------
 vc.vpx_event_arg       | 2026-03-25 02:00:01   | 2026-03-25 14:30:22  | 2026-03-25
02:00:01   |       2341
 vc.vpx_event           | 2026-03-25 02:00:01   | 2026-03-25 14:28:11  | 2026-03-25
02:00:01   |       1822
 ...
```

Manual Vacuum (if needed)

```
# Standard vacuum (non-blocking)
/opt/vmware/vpostgres/current/bin/psql -U postgres -d VCDB -c "VACUUM ANALYZE;"

# Full vacuum (blocking, reclaims space)
/opt/vmware/vpostgres/current/bin/psql -U postgres -d VCDB -c "VACUUM FULL
ANALYZE;"
```

Warning: `VACUUM FULL` acquires an exclusive lock on each table and rewrites the entire table. Only run during a maintenance window. Standard `VACUUM ANALYZE` is safe to run at any time.

Condition	Badge
Autovacuum ran in last 24h, dead tuples < 10,000	PASS
Autovacuum > 48h ago or dead tuples 10,000 - 100,000	WARN
No vacuum in 7+ days or dead tuples > 100,000	FAIL

8. vCenter HA (VCHA)

vCenter High Availability (VCHA) provides automated failover for vCenter Server using an active/passive/witness architecture.

8.1 VCHA Mode & Status

API Check

```
curl -sk -X GET \
  "https://${VC_FQDN}/api/vcenter/vcha/cluster/mode" \
  -H "vmware-api-session-id: ${VC_TOKEN}" | jq .
```

Expected Output (VCHA Enabled):

```
{
  "mode": "ENABLED"
}
```

Get Full VCHA Cluster Status

```
curl -sk -X POST \
  "https://${VC_FQDN}/api/vcenter/vcha/cluster?action=get" \
  -H "vmware-api-session-id: ${VC_TOKEN}" \
  -H "Content-Type: application/json" \
  -d '{"partial": false}' | jq .
```

Expected Output (Healthy VCHA):

```
{
  "config_state": "CONFIGURED",
  "mode": "ENABLED",
  "health_state": "HEALTHY",
  "node1": {
    "state": "UP",
```

```
      "role": "ACTIVE",
      "runtime": {
        "ip": {
          "ipv4": { "address": "10.0.0.101" }
        }
      }
    },
    "node2": {
      "state": "UP",
      "role": "PASSIVE",
      "runtime": {
        "ip": {
          "ipv4": { "address": "10.0.0.102" }
        }
      }
    },
    "witness": {
      "state": "UP",
      "runtime": {
        "ip": {
          "ipv4": { "address": "10.0.0.103" }
        }
      }
    }
  }
```

Condition	Badge
mode=ENABLED, health_state=HEALTHY, all nodes UP	PASS
mode=ENABLED but one node degraded	WARN
mode=DISABLED or health_state not HEALTHY	FAIL

8.2 Active / Passive / Witness Status

CLI Check (from Active Node)

```
# Check VCHA state via CLI
/usr/lib/vmware-vcha/vcha-cli cluster-get-state
```

Expected Output:

```
VCHA Cluster State: HEALTHY
Active Node State: UP
Passive Node State: UP
Witness Node State: UP
Replication State: IN_SYNC
```

Database Replication Lag

```
# Check replication lag from active node
/opt/vmware/vpostgres/current/bin/psql -U postgres -c \
  "SELECT client_addr, state, sent_lsn, write_lsn, flush_lsn, replay_lsn,
   (sent_lsn - replay_lsn) AS replication_lag
   FROM pg_stat_replication;"
```

Condition	Badge
Replication state IN_SYNC, lag = 0	PASS
Replication active but lag > 0	WARN
Replication not running	FAIL

8.3 Failover Readiness

Manual Failover Test (Planned)

Warning: Only perform planned failover during a maintenance window. This will temporarily disconnect all vSphere Client sessions.

```
# Initiate planned failover via API
curl -sk -X POST \
  "https://${VC_FQDN}/api/vcenter/vcha/cluster?action=failover" \
  -H "vmware-api-session-id: ${VC_TOKEN}" \
  -H "Content-Type: application/json" \
  -d '{"planned": true}'
```

Remediation (VCHA Degraded):
1. Check network connectivity between all three VCHA nodes (HA network)

2. Verify passive node is reachable: `ping <passive-ip>`
3. Verify witness node is reachable: `ping <witness-ip>`
4. Check VCHA logs: `/var/log/vmware/vcha/vcha.log`
5. If passive node offline, redeploy: remove and re-add VCHA via vSphere Client

9. Certificate Health

Certificate expiry is one of the most common causes of vCenter outages. Regular monitoring of all certificate stores is essential.

9.1 Certificate Manager Tool

```
# Launch Certificate Manager (interactive)
/usr/lib/vmware-vmca/bin/certificate-manager
```

This interactive tool provides options:

1. Replace Machine SSL certificate with Custom Certificate
2. Replace VMCA Root certificate with Custom Signing Certificate
3. Replace Machine SSL certificate with VMCA Certificate
4. Regenerate a new VMCA Root Certificate
5. Replace Solution user certificates with Custom Certificate
6. Replace Solution user certificates with VMCA certificates
7. Revert last performed operation
8. Reset all Certificates

9.2 VECS Store Listing

List All VECS Stores

```
/usr/lib/vmware-vmafd/bin/vecs-cli store list
```

Expected Output:

```
MACHINE_SSL_CERT
TRUSTED_ROOTS
TRUSTED_ROOT_CRLS
machine
vsphere-webclient
vpxd
vpxd-extension
hvc
data-encipherment
```

```
APPLMGMT_PASSWORD
SMS
wcp
backup_store
```

List Certificates in a Store

```
# List Machine SSL certificate
/usr/lib/vmware-vmafd/bin/vecs-cli entry list --store MACHINE_SSL_CERT
```

Expected Output:

```
Alias : __MACHINE_CERT
Entry type : Private Key
```

Get Certificate Details from a Store

```
/usr/lib/vmware-vmafd/bin/vecs-cli entry getcert \
  --store MACHINE_SSL_CERT --alias __MACHINE_CERT | \
  openssl x509 -noout -subject -issuer -dates -serial
```

Expected Output:

```
subject=CN = vcenter01.lab.local
issuer=CN = CA, DC = vsphere, DC = local, C = US, ST = California, O = lab.local
notBefore=Jan 15 00:00:00 2026 GMT
notAfter=Jan 15 00:00:00 2028 GMT
serial=3A4B5C6D7E8F
```

9.3 STS Certificate

The Security Token Service (STS) signing certificate is critical for SSO authentication.

Check STS Certificate Expiry

```
# Extract and check the STS signing certificate
/usr/lib/vmware-vmafd/bin/dir-cli trustedcert list \
```

```
  --login administrator@vsphere.local \
  --password "${VC_PASS}" | head -20
```

Alternative: Check via Lookup Service

```
# Get STS certificate from LDAP
/usr/lib/vmware-vmdir/bin/ldapsearch -h localhost -p 389 \
  -b "cn=TenantCredential-
1,cn=local,cn=Tenants,cn=IdentityManager,cn=Services,dc=vsphere,dc=local" \
  -D "cn=administrator,cn=users,dc=vsphere,dc=local" \
  -w "${VC_PASS}" \
  userCertificate 2>/dev/null | grep -A1 "userCertificate"
```

Check STS Token Signing Certificate with Python Script

```
# VMware-provided STS cert check script
python /usr/lib/vmware-vmca/share/config/checksts.py
```

Expected Output (Healthy):

```
STS signing certificate:
  Subject: CN=ssoserver-sign
  Not Before: Jan 15 00:00:00 2026 GMT
  Not After:  Jan 15 00:00:00 2028 GMT
  Days remaining: 661
  Status: VALID
```

Condition	Badge
STS cert > 60 days until expiry	PASS
STS cert 30 - 60 days until expiry	WARN
STS cert < 30 days or expired	FAIL

9.4 Machine SSL Certificate

```
# Check Machine SSL cert expiry remotely
echo | openssl s_client -connect ${VC_FQDN}:443 -servername ${VC_FQDN}
2>/dev/null | \
  openssl x509 -noout -subject -issuer -dates -checkend 2592000
```

Expected Output (Healthy):

```
subject=CN = vcenter01.lab.local
issuer=CN = CA, DC = vsphere, DC = local
notBefore=Jan 15 00:00:00 2026 GMT
notAfter=Jan 15 00:00:00 2028 GMT
Certificate will not expire
```

Check Expiry of All Solution User Certificates

```
for store in machine vsphere-webclient vpxd vpxd-extension; do
  echo "=== Store: ${store} ==="
  /usr/lib/vmware-vmafd/bin/vecs-cli entry getcert \
    --store ${store} --alias ${store} 2>/dev/null | \
    openssl x509 -noout -subject -dates 2>/dev/null
done
```

9.5 Expiry Checks -- All Certificates

Comprehensive Certificate Expiry Report

```
# Check all VECS stores for certificate expiry
for store in $(/usr/lib/vmware-vmafd/bin/vecs-cli store list); do
  for alias in $(/usr/lib/vmware-vmafd/bin/vecs-cli entry list --store ${store}
2>/dev/null | grep "Alias" | awk '{print $3}'); do
    CERT=$(/usr/lib/vmware-vmafd/bin/vecs-cli entry getcert --store ${store} --
alias ${alias} 2>/dev/null)
    if [ -n "$CERT" ]; then
      EXPIRY=$(echo "$CERT" | openssl x509 -noout -enddate 2>/dev/null | cut -d=
-f2)
      DAYS=$(echo "$CERT" | openssl x509 -noout -checkend 0 2>/dev/null && echo
"VALID" || echo "EXPIRED")
```

```
      printf "Store: %-25s Alias: %-25s Expires: %-30s Status: %s\n" "$store"
"$alias" "$EXPIRY" "$DAYS"
    fi
  done
done
```

API-based Certificate Check

```
curl -sk -X GET \
  "https://${VC_FQDN}/api/vcenter/certificate-management/vcenter/tls" \
  -H "vmware-api-session-id: ${VC_TOKEN}" | jq '{
    subject: .subject_dn,
    issuer: .issuer_dn,
    valid_from: .valid_from,
    valid_to: .valid_to
  }'
```

Expected Output:

```
{
  "subject": "CN=vcenter01.lab.local",
  "issuer": "CN=CA, DC=vsphere, DC=local",
  "valid_from": "2026-01-15T00:00:00.000Z",
  "valid_to": "2028-01-15T00:00:00.000Z"
}
```

Condition	Badge
All certificates > 60 days until expiry	PASS
Any certificate 30 - 60 days until expiry	WARN
Any certificate < 30 days or expired	FAIL

Remediation (Certificate Expiring/Expired):
1. For Machine SSL: Use Certificate Manager option 3 or 1 to replace
2. For STS: Use `/usr/lib/vmware-vmca/bin/certificate-manager` option 8 (reset) as last resort
3. For Solution Users: Use Certificate Manager option 6 to regenerate with VMCA
4. KB Reference: KB 2111411 for STS certificate renewal

10. Storage Health

10.1 Disk Partitions & Filesystem

Check All Partitions

```
df -h
```

Expected Output (Healthy):

```
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda3        11G  4.2G  6.1G  41% /
tmpfs           6.0G   48M  5.9G   1% /dev/shm
/dev/sda1       128M   32M   97M  25% /boot
/dev/sda5        25G  5.8G   18G  25% /storage/log
/dev/sda6        10G  2.1G  7.4G  22% /storage/db
/dev/sda8        50G  9.2G   38G  20% /storage/seat
/dev/sda9        25G  3.3G   20G  15% /storage/netdump
/dev/sda10       10G  1.2G  8.2G  13% /storage/autodeploy
/dev/sda11       10G  836M  8.6G   9% /storage/imagebuilder
/dev/sda12       10G  2.5G  7.0G  27% /storage/updatemgr
/dev/sda13        5G   63M  4.7G   2% /storage/lifecycle
```

Check inode Usage

```
df -ih
```

Partition	Warn Threshold	Fail Threshold	Badge (Healthy)
`/` (root)	> 70%	> 85%	PASS
`/storage/log`	> 70%	> 85%	PASS
`/storage/db`	> 70%	> 85%	PASS
`/storage/seat`	> 70%	> 85%	PASS
All others	> 75%	> 90%	PASS

10.2 Log Storage Utilization

```
# Check /storage/log usage
du -sh /storage/log/* 2>/dev/null | sort -rh | head -15
```

Expected Output:

```
2.1G    /storage/log/vmware/vpxd
812M    /storage/log/vmware/vsphere-ui
543M    /storage/log/vmware/sso
322M    /storage/log/vmware/eam
210M    /storage/log/vmware/rhttpproxy
...
```

Find Large Log Files

```
find /storage/log -type f -size +100M -exec ls -lh {} \; 2>/dev/null
```

10.3 DB Storage Utilization

```
# Check /storage/db usage
du -sh /storage/db/*
```

Expected Output:

```
8.2G    /storage/db/vpostgres
```

```
# Check PostgreSQL WAL files
du -sh /storage/db/vpostgres/pg_wal/
```

Condition	Badge
All partitions < 70%	PASS

Any partition 70 - 85%	WARN
Any partition > 85%	FAIL

Remediation (Storage Full):
1. Clear old logs: `find /storage/log -name "*.log" -mtime +7 -delete`
2. Rotate logs: `logrotate -f /etc/logrotate.conf`
3. Clean temp files: `rm -rf /storage/log/vmware/vpxd/vpxd-*.log.[0-9]*`
4. Purge old WAL files: `/opt/vmware/vpostgres/current/bin/pg_archivecleanup /storage/db/vpostgres/pg_wal/ <oldest_needed_wal>`
5. Expand disk: Shutdown appliance, expand VMDK, boot, run `/usr/lib/applmgmt/support/scripts/expand_disk.sh`

11. Performance & Resource Utilization

11.1 CPU Utilization

API Check

```
curl -sk -X GET \
  "https://${VC_FQDN}/api/appliance/health/load" \
  -H "vmware-api-session-id: ${VC_TOKEN}"
```

CLI Check

```
# Current CPU load
uptime
```

Expected Output:

```
14:22:33 up 3 days,  4:07,  1 user,  load average: 1.23, 1.45, 1.32
```

```
# Top CPU consumers
top -bn1 | head -20
```

```
# CPU info
nproc
cat /proc/cpuinfo | grep "model name" | head -1
```

Condition	Badge
Load average < number of CPUs (< 70% per core)	PASS
Load average 70-85% of CPU count	WARN
Load average > 85% of CPU count sustained	FAIL

11.2 Memory Utilization

API Check

```
curl -sk -X GET \
  "https://${VC_FQDN}/api/appliance/health/mem" \
  -H "vmware-api-session-id: ${VC_TOKEN}"
```

CLI Check

```
free -m
```

Expected Output (Healthy):

```
               total        used        free      shared  buff/cache   available
Mem:           24576       14234        2180         312        8162        9876
Swap:           3071          32        3039
```

```
# Memory usage percentage
free -m | awk 'NR==2{printf "Memory Usage: %.1f%%\n", $3*100/$2}'
```

Condition	Badge
Memory usage < 80%	PASS
Memory usage 80 - 90%	WARN
Memory usage > 90%	FAIL

11.3 Swap & Load Average

```
# Swap usage
swapon --show
```

Expected Output (Healthy):

```
NAME      TYPE SIZE USED PRIO
/dev/sda2 partition 3G  32M   -2
```

```
# Detailed swap info
cat /proc/swaps
vmstat 1 5
```

Condition	Badge
Swap usage < 5%	PASS
Swap usage 5 - 25%	WARN
Swap usage > 25%	FAIL

Remediation (Performance Degradation):
1. Identify top consumers: `top -bn1 -o %MEM | head -20`
2. Restart heavy service: `service-control --stop vmware-vpxd && service-control --start vmware-vpxd`
3. Check for runaway Java processes: `ps aux | grep java | grep -v grep`
4. Increase VM resources: Add more vCPUs or memory to the appliance VM
5. Check for DRS/HA tasks in loop: review `/var/log/vmware/vpxd/vpxd.log` for repeated task patterns

12. SSO / Identity Source Health

12.1 SSO Domain Health

Check SSO Domain Status

```
# List SSO domains
/opt/vmware/bin/sso-config.sh -get_identity_sources
```

Expected Output:

```
Identity Source: vsphere.local
Type: System Domain
Default: true

Identity Source: lab.local
Type: ActiveDirectory
Default: false
```

Verify SSO Configuration

```
/opt/vmware/bin/sso-config.sh -get_default_identity_sources
```

12.2 Identity Source Connectivity

API Check

```
curl -sk -X GET \
  "https://${VC_FQDN}/api/vcenter/identity/providers" \
  -H "vmware-api-session-id: ${VC_TOKEN}" | jq .
```

Test Authentication

```
# Test SSO login via API session creation
curl -sk -w "\nHTTP_CODE: %{http_code}\n" -X POST \
  "https://${VC_FQDN}/api/session" \
  -u "administrator@vsphere.local:${VC_PASS}"
```

Expected Output (Healthy):

```
"b1a2c3d4-e5f6-7890-abcd-ef1234567890"
HTTP_CODE: 201
```

Condition	Badge
HTTP 201, session token returned	PASS
HTTP 401 (credentials issue)	WARN
HTTP 500 or connection timeout	FAIL

12.3 LDAP / AD Binding Test

Test LDAP Connectivity

```
# Test LDAP bind to AD (if AD identity source configured)
ldapsearch -h dc01.lab.local -p 389 \
  -D "CN=svc_vcenter,OU=Service Accounts,DC=lab,DC=local" \
  -w 'ServiceAccountPassword' \
  -b "DC=lab,DC=local" \
  -s base "(objectClass=*)" 2>&1 | head -5
```

Expected Output (Healthy):

```
# extended LDIF
#
# LDAPv3
# base <DC=lab,DC=local> with scope baseObject
# filter: (objectClass=*)
```

Check VMware Directory Service (vmdir)

```
# Check vmdir service status
systemctl status vmware-stsd
/opt/vmware/bin/ldapsearch -h localhost -p 389 \
  -b "" -s base "(objectClass=*)" namingContexts 2>/dev/null
```

12.4 Token Validation

```
# Verify existing session token is valid
curl -sk -X GET \
  "https://${VC_FQDN}/api/session" \
  -H "vmware-api-session-id: ${VC_TOKEN}" | jq .
```

Expected Output (Valid Token):

```
{
  "user": "VSPHERE.LOCAL\\Administrator",
  "created_time": "2026-03-26T10:15:22.000Z"
}
```

Condition	Badge
Token valid, user details returned	PASS
Token expired (HTTP 401)	WARN
SSO service unreachable (HTTP 503)	FAIL

Remediation (SSO Issues):
1. Restart STS service: `service-control --stop vmware-stsd && service-control --start vmware-stsd`
2. Check STS logs: `tail -100 /var/log/vmware/sso/ssoAdminServer.log`
3. For AD connectivity: verify DNS, network, service account credentials
4. For lockout: unlock admin via `/usr/lib/vmware-vmdir/bin/dir-cli password reset --account administrator --new <new_pass>`
5. STS cert expiry: Replace using KB 79248 procedures

13. Plugins & Extensions

13.1 Registered Plugins

API Check

```
curl -sk -X GET \
  "https://${VC_FQDN}/api/vcenter/namespace-management/software/registries" \
  -H "vmware-api-session-id: ${VC_TOKEN}" 2>/dev/null | jq .
```

MOB (Managed Object Browser) Method

Navigate to: `https://$VC_FQDN/mob/?moid=ExtensionManager&doPath=extensionList`

PowerCLI Method

```
Connect-VIServer -Server $VC_FQDN -User $VC_USER -Password $VC_PASS
$em = Get-View ExtensionManager
$em.ExtensionList | Select-Object Key, Description, Company, Version | Format-
Table -AutoSize
```

Expected Output:

```
Key                                Description             Company
Version
---                                -----------             -------
-------
com.vmware.vim.sms                 Storage Monitoring      VMware, Inc.
8.0.3
com.vmware.vcIntegrity             vSphere Lifecycle Mgr   VMware, Inc.
8.0.3
com.vmware.vim.eam                 ESX Agent Manager       VMware, Inc.
8.0.3
com.vmware.rbd                     RBD                     VMware, Inc.
8.0.3
com.vmware.h4.vsphere.client       vSphere Client          VMware, Inc.
8.0.3
```

```
com.vmware.nsx.management.nsxt          NSX                         VMware, Inc.
4.2.1
...
```

13.2 Plugin Health Verification

```
# Check plugin health via REST
curl -sk -X GET \
  "https://${VC_FQDN}/ui/extensionmanager/extensionlist" \
  -H "vmware-api-session-id: ${VC_TOKEN}" 2>/dev/null | jq '.[].key'
```

Check for Plugin Load Errors

```
# Check vsphere-ui logs for plugin errors
grep -i "plugin\|extension" /var/log/vmware/vsphere-
ui/logs/vsphere_client_virgo.log | \
  grep -i "error\|fail\|exception" | tail -20
```

Condition	Badge
All expected plugins registered and loading	PASS
Some plugins failing to load but not critical	WARN
Core plugins missing or all failing	FAIL

13.3 Stale Plugin Cleanup

Warning: Only remove plugins that have been confirmed as stale (e.g., from decommissioned products). Removing active plugins will break functionality.

Identify Stale Plugins

```
# Check for plugins whose server URL is unreachable
curl -sk -X GET \
```

```
  "https://${VC_FQDN}/mob/?moid=ExtensionManager" \
  -H "vmware-api-session-id: ${VC_TOKEN}" 2>/dev/null
```

Remove Stale Plugin via MOB

1. Navigate to: `https://$VC_FQDN/mob/?moid=ExtensionManager`
2. Click `UnregisterExtension`
3. Enter the extension key (e.g., `com.vendor.stale.plugin`)
4. Click `Invoke Method`

Remove via PowerCLI

```
$em = Get-View ExtensionManager
$em.UnregisterExtension("com.vendor.stale.plugin")
```

Remediation (Plugin Issues):
1. Restart vSphere Client: `service-control --stop vsphere-ui && service-control --start vsphere-ui`
2. Clear client cache: `rm -rf /etc/vmware/vsphere-ui/cm-init-*`
3. Re-register plugin: reinstall the product that provides the plugin
4. Check plugin compatibility with current vCenter version

14. Lookup Service & PSC

The Lookup Service is the service registry for all vSphere components. Since vCenter 7.0+, the Platform Services Controller (PSC) is embedded.

14.1 Lookup Service Registration

Check Lookup Service Status

```
service-control --status vmware-lookupsvc
```

Expected Output:

```
VMware Lookup Service:Status: RUNNING
```

List All Service Registrations

```
# Use lstool to list registrations
python /usr/lib/vmidentity/tools/scripts/lstool.py list \
  --url "https://localhost/lookupservice/sdk" \
  --no-check-cert 2>/dev/null | head -60
```

Expected Output (Healthy):

```
=== Service Registration ===
Service ID: vcenterserver
Owner ID: vcenter01.lab.local@vsphere.local
Service Type: vcenterserver
Endpoints:
  URL: https://vcenter01.lab.local/sdk
  Protocol: vmomi

Service ID: cs.identity
Owner ID: vcenter01.lab.local@vsphere.local
Service Type: cs.identity
Endpoints:
  URL: https://vcenter01.lab.local/sts/STSService/vsphere.local
```

```
  Protocol: wsTrust
...
```

14.2 STS Health

```
# Check STS service status
service-control --status vmware-stsd
```

Expected Output:

```
VMware Security Token Service:Status: RUNNING
```

Test STS Token Issuance

```
# Verify STS can issue tokens by creating an API session
curl -sk -X POST \
  "https://${VC_FQDN}/api/session" \
  -u "${VC_USER}:${VC_PASS}" \
  -w "\nHTTP Status: %{http_code}\n"
```

Condition	Badge
STS running, tokens issued successfully	PASS
STS running but slow token issuance (> 5s)	WARN
STS stopped or tokens not issued	FAIL

14.3 Service Registration Entries

Verify Key Registrations Exist

```
python /usr/lib/vmidentity/tools/scripts/lstool.py list \
  --url "https://localhost/lookupservice/sdk" \
  --no-check-cert 2>/dev/null | grep "Service Type" | sort -u
```

Expected Service Types:

```
Service Type: cs.authorization
Service Type: cs.identity
Service Type: cs.license
Service Type: cs.lookup
Service Type: cs.privilege
Service Type: sso:admin
Service Type: sso:groupcheck
Service Type: sso:sts
Service Type: vcenterserver
Service Type: cs.inventory
Service Type: cs.envoy
```

Condition	Badge
All expected service types registered	PASS
Some non-critical registrations missing	WARN
Core registrations (sts, identity, vcenterserver) missing	FAIL

Remediation (Lookup Service Issues):
1. Restart Lookup Service: `service-control --stop vmware-lookupsvc && service-control --start vmware-lookupsvc`
2. Re-register services: `/usr/lib/vmware-lookupsvc/tools/ls_update_certs.py --url https://localhost/lookupservice/sdk --fingerprint <thumbprint>`
3. Check logs: `/var/log/vmware/lookupsvc/lookupsvc.log`
4. If corrupted, run: `/usr/lib/vmware-lookupsvc/tools/ls_recover.py`

15. Inventory Verification

15.1 Datacenter / Cluster / Host / VM Counts

Get Datacenter Count

```
curl -sk -X GET \
  "https://${VC_FQDN}/api/vcenter/datacenter" \
  -H "vmware-api-session-id: ${VC_TOKEN}" | jq 'length'
```

List All Datacenters

```
curl -sk -X GET \
  "https://${VC_FQDN}/api/vcenter/datacenter" \
  -H "vmware-api-session-id: ${VC_TOKEN}" | jq '.[] | {name, datacenter}'
```

Expected Output:

```
{
  "name": "DC-Site-A",
  "datacenter": "datacenter-1"
}
{
  "name": "DC-Site-B",
  "datacenter": "datacenter-2"
}
```

Get Cluster Count and List

```
curl -sk -X GET \
  "https://${VC_FQDN}/api/vcenter/cluster" \
```

```
  -H "vmware-api-session-id: ${VC_TOKEN}" | jq '.[] | {name, cluster, ha_enabled,
drs_enabled}'
```

Expected Output:

```
{
  "name": "Management-Cluster",
  "cluster": "domain-c8",
  "ha_enabled": true,
  "drs_enabled": true
}
{
  "name": "Workload-Cluster-01",
  "cluster": "domain-c44",
  "ha_enabled": true,
  "drs_enabled": true
}
```

Get Host Count and Status

```
curl -sk -X GET \
  "https://${VC_FQDN}/api/vcenter/host" \
  -H "vmware-api-session-id: ${VC_TOKEN}" | jq '.[] | {name, host,
connection_state, power_state}'
```

Expected Output:

```
{
  "name": "esxi01.lab.local",
  "host": "host-10",
  "connection_state": "CONNECTED",
  "power_state": "POWERED_ON"
}
{
  "name": "esxi02.lab.local",
  "host": "host-11",
  "connection_state": "CONNECTED",
```

```
  "power_state": "POWERED_ON"
}
```

Get VM Count

```
# Total VM count
curl -sk -X GET \
  "https://${VC_FQDN}/api/vcenter/vm" \
  -H "vmware-api-session-id: ${VC_TOKEN}" | jq 'length'
```

```
# Powered-on VMs
curl -sk -X GET \
  "https://${VC_FQDN}/api/vcenter/vm?power_states=POWERED_ON" \
  -H "vmware-api-session-id: ${VC_TOKEN}" | jq 'length'
```

Full Inventory Summary Script

```
echo "=== vCenter Inventory Summary ==="
DC=$(curl -sk "https://${VC_FQDN}/api/vcenter/datacenter" -H "vmware-api-session-id: ${VC_TOKEN}" | jq 'length')
CL=$(curl -sk "https://${VC_FQDN}/api/vcenter/cluster" -H "vmware-api-session-id: ${VC_TOKEN}" | jq 'length')
HO=$(curl -sk "https://${VC_FQDN}/api/vcenter/host" -H "vmware-api-session-id: ${VC_TOKEN}" | jq 'length')
VM=$(curl -sk "https://${VC_FQDN}/api/vcenter/vm" -H "vmware-api-session-id: ${VC_TOKEN}" | jq 'length')
VMon=$(curl -sk "https://${VC_FQDN}/api/vcenter/vm?power_states=POWERED_ON" -H "vmware-api-session-id: ${VC_TOKEN}" | jq 'length')
DS=$(curl -sk "https://${VC_FQDN}/api/vcenter/datastore" -H "vmware-api-session-id: ${VC_TOKEN}" | jq 'length')
NET=$(curl -sk "https://${VC_FQDN}/api/vcenter/network" -H "vmware-api-session-id: ${VC_TOKEN}" | jq 'length')
echo "Datacenters:  ${DC}"
echo "Clusters:     ${CL}"
echo "Hosts:        ${HO}"
echo "Total VMs:    ${VM}"
echo "Powered-On:   ${VMon}"
```

```
echo "Datastores:   ${DS}"
echo "Networks:     ${NET}"
```

15.2 Inventory Consistency

Check for Disconnected Hosts

```
curl -sk -X GET \
  "https://${VC_FQDN}/api/vcenter/host?
connection_states=DISCONNECTED,NOT_RESPONDING" \
  -H "vmware-api-session-id: ${VC_TOKEN}" | jq '.[] | {name, connection_state}'
```

Expected Output (Healthy): Empty array `[]`

Check for Orphaned VMs

```
curl -sk -X GET \
  "https://${VC_FQDN}/api/vcenter/vm?power_states=POWERED_OFF" \
  -H "vmware-api-session-id: ${VC_TOKEN}" | jq '.[] | {name, vm, power_state}'
```

Condition	Badge
All hosts CONNECTED, inventory counts match expected	PASS
Some hosts in maintenance mode (planned)	WARN
Disconnected/NOT_RESPONDING hosts found	FAIL

Remediation (Inventory Issues):
1. Reconnect host: Right-click host in vSphere Client > Connection > Connect
2. Via API: `curl -sk -X POST "https://${VC_FQDN}/api/vcenter/host/host-XX?action=connect" -H "vmware-api-session-id: ${VC_TOKEN}"`
3. Remove orphaned objects: Right-click > Remove from Inventory
4. If hosts persistently disconnect, check network, DNS, and host certificates

16. Syslog & Log Configuration

16.1 Syslog Forwarding

Check Syslog Configuration via API

```
curl -sk -X GET \
  "https://${VC_FQDN}/api/appliance/logging/forwarding" \
  -H "vmware-api-session-id: ${VC_TOKEN}" | jq .
```

Expected Output (Configured):

```
[
  {
    "hostname": "syslog.lab.local",
    "port": 514,
    "protocol": "UDP"
  },
  {
    "hostname": "loginsight.lab.local",
    "port": 9000,
    "protocol": "TCP"
  }
]
```

Test Syslog Forwarding

```
curl -sk -X POST \
  "https://${VC_FQDN}/api/appliance/logging/forwarding?action=test" \
  -H "vmware-api-session-id: ${VC_TOKEN}" \
  -H "Content-Type: application/json" \
  -d '{"send_test_message": true}' | jq .
```

Expected Output:

```
[
  {
    "hostname": "syslog.lab.local",
    "state": "UP",
    "message": ""
  }
]
```

Configure Syslog Forwarding

```
curl -sk -X PUT \
  "https://${VC_FQDN}/api/appliance/logging/forwarding" \
  -H "vmware-api-session-id: ${VC_TOKEN}" \
  -H "Content-Type: application/json" \
  -d '[{"hostname": "syslog.lab.local", "port": 514, "protocol": "UDP"}]'
```

Condition	Badge
Syslog configured and test passes (state=UP)	PASS
Syslog configured but test fails	WARN
Syslog not configured	FAIL

16.2 Log Rotation

Check Logrotate Configuration

```
# Check logrotate status
cat /etc/logrotate.conf | head -20

# Check vCenter-specific rotation configs
ls -la /etc/logrotate.d/
```

Trigger Manual Log Rotation

```
logrotate -f /etc/logrotate.conf
```

16.3 Log Bundle Generation

Generate Support Bundle via API

```
curl -sk -X POST \
  "https://${VC_FQDN}/api/appliance/support-bundle" \
  -H "vmware-api-session-id: ${VC_TOKEN}" \
  -H "Content-Type: application/json" | jq .
```

Generate via CLI

```
# Generate support bundle
/usr/lib/vmware-vpxd/support/vcdb_report.sh

# Generate full log bundle
vc-support-bundle
```

Tip: Support bundles can be large (several GB). Ensure sufficient free space in `/storage/log` before generating. Target directory: `/var/log/vmware/support/`

17. NTP Configuration

Time synchronization is critical for vCenter operations, certificate validation, SSO token integrity, and log correlation.

17.1 Time Sync via API

Get NTP Configuration

```
curl -sk -X GET \
  "https://${VC_FQDN}/api/appliance/ntp" \
  -H "vmware-api-session-id: ${VC_TOKEN}" | jq .
```

Expected Output:

```
[
  "ntp1.lab.local",
  "ntp2.lab.local"
]
```

Get Time Sync Mode

```
curl -sk -X GET \
  "https://${VC_FQDN}/api/appliance/timesync" \
  -H "vmware-api-session-id: ${VC_TOKEN}"
```

Expected Output:

```
"NTP"
```

Set NTP Servers via API

```
curl -sk -X PUT \
  "https://${VC_FQDN}/api/appliance/ntp" \
  -H "vmware-api-session-id: ${VC_TOKEN}" \
```

```
  -H "Content-Type: application/json" \
  -d '["ntp1.lab.local", "ntp2.lab.local"]'
```

17.2 Time Sync via CLI

Check NTP Service

```
# Check NTP daemon status
systemctl status systemd-timesyncd
```

```
# Check NTP synchronization
timedatectl status
```

Expected Output (Healthy):

```
               Local time: Thu 2026-03-26 14:22:33 UTC
           Universal time: Thu 2026-03-26 14:22:33 UTC
                 RTC time: Thu 2026-03-26 14:22:33
                Time zone: UTC (UTC, +0000)
System clock synchronized: yes
              NTP service: active
          RTC in local TZ: no
```

```
# Check NTP peers
ntpq -p 2>/dev/null || chronyc sources 2>/dev/null
```

Expected Output:

```
MS Name/IP address         Stratum Poll Reach LastRx Last sample
===============================================================================
^* ntp1.lab.local                2   6   377    34   +0.234ms[+0.312ms] +/- 12ms
^+ ntp2.lab.local                2   6   377    35   +1.023ms[+1.101ms] +/- 15ms
```

17.3 Drift Check

```
# Check clock offset
chronyc tracking 2>/dev/null | grep "System time\|Last offset"
```

Expected Output:

```
System time     : 0.000000234 seconds fast of NTP time
Last offset     : +0.000000312 seconds
```

Condition	Badge
NTP configured, synced, drift < 1 second	PASS
NTP configured but drift 1 - 5 seconds	WARN
NTP not configured or drift > 5 seconds	FAIL

Remediation (NTP Issues):
1. Set NTP servers: `curl -sk -X PUT "https://${VC_FQDN}/api/appliance/ntp" -H "vmware-api-session-id: ${VC_TOKEN}" -H "Content-Type: application/json" -d '["pool.ntp.org"]'`
2. Force time sync: `systemctl restart systemd-timesyncd`
3. Verify mode is NTP (not HOST): `curl -sk -X PUT "https://${VC_FQDN}/api/appliance/timesync" -H "vmware-api-session-id: ${VC_TOKEN}" -H "Content-Type: application/json" -d '"NTP"'`
4. Check firewall allows NTP (UDP 123) outbound

18. Port Reference Table

vCenter Server Inbound Ports

Port	Protocol	Service	Source	Description
22	TCP	SSH	Admin workstations	Appliance shell access (should be disabled in production)
80	TCP	HTTP	All clients	Redirects to HTTPS (443)
443	TCP	HTTPS	All clients	vSphere Client, REST API, SDK, MOB
389	TCP	LDAP	PSC components	VMware Directory Service (vmdir)
636	TCP	LDAPS	PSC components	VMware Directory Service (secure)
902	TCP/UDP	VMware Auth	ESXi hosts	VM console proxy, host management
1514	TCP	Syslog (TLS)	ESXi hosts	Syslog collection from hosts
2012	TCP	Control Interface	Internal	vCenter control interface
2020	TCP	Auth Framework	Internal	Authentication framework
5480	TCP	VAMI	Admin workstations	Appliance Management Interface
6501	TCP	Auto Deploy	ESXi hosts	Auto Deploy service
6502	TCP	Auto Deploy	ESXi hosts	Auto Deploy reverse proxy
7080	TCP	Secure Token	Internal	VMware STS (HTTP)
7444	TCP	Secure Token	Internal	VMware STS (HTTPS)
8084	TCP	Update Manager	ESXi hosts	vSphere Update Manager
9084	TCP	Update Manager	ESXi hosts	Update Manager web client
9087	TCP	Analytics	Internal	Analytics service
9123	TCP	Migration Assistant	External	vCenter migration

vCenter Server Outbound Ports

Port	Protocol	Destination	Description
53	TCP/UDP	DNS servers	DNS resolution

88	TCP/UDP	AD/KDC	Kerberos authentication
123	UDP	NTP servers	Time synchronization
389	TCP	AD/LDAP	Identity source queries
443	TCP	ESXi hosts	Host management via HTTPS
443	TCP	NSX Manager	NSX integration
443	TCP	SDDC Manager	VCF lifecycle management
514	UDP	Syslog server	Syslog forwarding
902	TCP	ESXi hosts	VM console, host management

Port Verification Commands

```
# Check listening ports on vCenter appliance
ss -tlnp | sort -t: -k2 -n
```

```
# Check specific port connectivity
curl -sk -o /dev/null -w "%{http_code}" https://${VC_FQDN}:443
curl -sk -o /dev/null -w "%{http_code}" https://${VC_FQDN}:5480
```

```
# Check outbound connectivity
nc -zv ntp1.lab.local 123 2>&1
nc -zv dc01.lab.local 389 2>&1
nc -zv esxi01.lab.local 443 2>&1
```

19. Common Issues & Remediation

19.1 VPXD Crashes or Won't Start

Symptoms: vSphere Client inaccessible, `service-control --status vmware-vpxd` shows STOPPED.

Impact: Complete vCenter management outage. VMs continue running on ESXi hosts but cannot be managed.

Diagnostic Steps:

```
# Check VPXD logs for crash reason
tail -200 /var/log/vmware/vpxd/vpxd.log | grep -i "error\|fatal\|abort"

# Check for core dumps
ls -la /var/core/

# Check if database is reachable
systemctl status vmware-vpostgres

# Check disk space (VPXD won't start if disk full)
df -h /storage/log /storage/db /

# Check for port conflicts
ss -tlnp | grep ":443\b"
```

Remediation:
1. If disk full: clear logs per Section 10, then start VPXD
2. If DB down: `service-control --start vmware-vpostgres`, then start VPXD
3. If port conflict: identify and stop conflicting process, then start VPXD
4. If persistent crash: collect support bundle and open VMware SR
5. As last resort: `service-control --stop --all && service-control --start --all`

19.2 Database Corruption

Symptoms: VPXD errors referencing VCDB, inventory inconsistencies, SQL errors in logs.

```
# Check PostgreSQL logs
tail -100 /var/log/vmware/vpostgres/postgresql*.log
```

```
# Run database integrity check
/opt/vmware/vpostgres/current/bin/pg_isready -h localhost -p 5432

# Check for corruption indicators
/opt/vmware/vpostgres/current/bin/psql -U postgres -d VCDB -c \
  "SELECT datname, datallowconn FROM pg_database;"
```

Remediation:
1. If minor: Run `VACUUM FULL ANALYZE;` to reclaim and rebuild
2. If tables corrupted: `REINDEX DATABASE VCDB;`
3. If severe: Restore from backup: `/opt/vmware/vpostgres/current/bin/pg_restore`
4. If no backup available: Contact VMware Global Support immediately
5. Prevention: Ensure regular file-based backups are configured

19.3 Certificate Expiry

Symptoms: SSO login failures, "certificate expired" errors in browser, service communication failures.

```
# Quick check all certs
for store in MACHINE_SSL_CERT machine vsphere-webclient vpxd vpxd-extension; do
  echo "=== ${store} ==="
  /usr/lib/vmware-vmafd/bin/vecs-cli entry getcert --store ${store} \
    --alias $(/usr/lib/vmware-vmafd/bin/vecs-cli entry list --store ${store}
2>/dev/null | grep Alias | awk '{print $3}' | head -1) 2>/dev/null | \
    openssl x509 -noout -dates 2>/dev/null
done

# Check STS cert specifically
python /usr/lib/vmware-vmca/share/config/checksts.py 2>/dev/null
```

Remediation:
1. For Machine SSL: `/usr/lib/vmware-vmca/bin/certificate-manager` option 3 (VMCA) or 1 (custom)
2. For STS expired: Follow KB 79248 to replace STS signing certificate
3. For Solution User certs: Certificate Manager option 6
4. After renewal: restart all services `service-control --stop --all && service-control --start --all`
5. Schedule proactive monitoring: check certs monthly

19.4 SSO Lockout

Symptoms: Cannot log in as administrator@vsphere.local, "invalid credentials" or "account locked."

```
# Check lockout policy
/opt/vmware/bin/sso-config.sh -get_lockout_policy

# Check failed login attempts
grep -i "login\|auth\|lock" /var/log/vmware/sso/ssoAdminServer.log | tail -30
```

Remediation:
1. Wait for lockout period to expire (default: 5 minutes)
2. Unlock via CLI: `/usr/lib/vmware-vmdir/bin/dir-cli account unlock --account administrator --password <current_password>`
3. Reset password: `/usr/lib/vmware-vmdir/bin/dir-cli password reset --account administrator --new <new_password>`
4. If vmdir corrupted: Boot to single-user mode and reset via `/usr/lib/vmware-vmdir/bin/vdcadmintool`
5. As absolute last resort: restore from backup

19.5 Performance Degradation

Symptoms: vSphere Client slow, API timeouts, tasks taking excessively long.

```
# Identify bottleneck
echo "=== CPU ===" && uptime
echo "=== Memory ===" && free -m
echo "=== Swap ===" && swapon --show
echo "=== Disk I/O ===" && iostat -x 1 3 2>/dev/null || echo "iostat not
available"
echo "=== Top Processes ===" && top -bn1 | head -15
echo "=== DB Connections ===" && /opt/vmware/vpostgres/current/bin/psql -U
postgres -d VCDB -c \
  "SELECT count(*) FROM pg_stat_activity;"
```

Remediation:
1. If CPU high: check for runaway tasks, reduce DRS sensitivity
2. If memory high: restart VPXD to reclaim, or add VM memory
3. If swap heavy: increase VM memory, check for memory leaks
4. If DB connections high: restart VPXD, check for stuck tasks
5. If disk I/O: move vCenter to faster storage (SSD/NVMe)
6. Long-term: right-size vCenter appliance per VMware sizing guidelines

20. CLI Quick Reference Card

Service Management

Command	Description
`service-control --status --all`	Show status of all vCenter services
`service-control --status vmware-vpxd`	Show VPXD service status
`service-control --start --all`	Start all vCenter services
`service-control --stop --all`	Stop all vCenter services
`service-control --start <service>`	Start a specific service
`service-control --stop <service>`	Stop a specific service
`vmon-cli --list`	List all vMon-managed services
`vmon-cli --status <service>`	Check specific vMon service status
`vmon-cli --start <service>`	Start a vMon service
`vmon-cli --stop <service>`	Stop a vMon service
`vmon-cli --restart <service>`	Restart a vMon service

System & Appliance

Command	Description
`uptime`	System uptime and load average
`free -m`	Memory usage in MB
`df -h`	Disk partition usage
`top -bn1`	Process listing (batch mode, single iteration)
`timedatectl status`	Time synchronization status
`hostnamectl`	Hostname and OS information
`cat /etc/applmgmt/appliance/update.conf`	Appliance version info
`vpxd -v`	VPXD version
`cat /etc/issue`	Photon OS version

Database

Command	Description
`/opt/vmware/vpostgres/current/bin/psql -U postgres -d VCDB`	Connect to VCDB
`systemctl status vmware-vpostgres`	PostgreSQL service status
`VACUUM ANALYZE;` (in psql)	Standard vacuum with analyze
`VACUUM FULL ANALYZE;` (in psql)	Full vacuum (blocking)
`SELECT pg_size_pretty(pg_database_size('VCDB'));`	VCDB size

Certificates

Command	Description
`/usr/lib/vmware-vmca/bin/certificate-manager`	Certificate Manager (interactive)
`/usr/lib/vmware-vmafd/bin/vecs-cli store list`	List VECS stores
`/usr/lib/vmware-vmafd/bin/vecs-cli entry list --store <store>`	List entries in a VECS store
`/usr/lib/vmware-vmafd/bin/vecs-cli entry getcert --store <store> --alias <alias>`	Get certificate from store
`/usr/lib/vmware-vmafd/bin/dir-cli trustedcert list`	List trusted root certificates
`python /usr/lib/vmware-vmca/share/config/checksts.py`	Check STS certificate

SSO & Identity

Command	Description
`/opt/vmware/bin/sso-config.sh -get_identity_sources`	List identity sources
`/opt/vmware/bin/sso-config.sh -get_default_identity_sources`	Get default identity source
`/opt/vmware/bin/sso-config.sh -get_lockout_policy`	SSO lockout policy
`/usr/lib/vmware-vmdir/bin/dir-cli password reset --account <user> --new <pass>`	Reset SSO password
`/usr/lib/vmware-vmdir/bin/dir-cli account unlock --account <user>`	Unlock SSO account

Lookup Service

Command	Description
`python /usr/lib/vmidentity/tools/scripts/lstool.py list --url https://localhost/lookupservice/sdk --no-check-cert`	List all registrations

Logs

Command	Description
`tail -f /var/log/vmware/vpxd/vpxd.log`	Follow VPXD log
`tail -f /var/log/vmware/sso/ssoAdminServer.log`	Follow SSO log
`tail -f /var/log/vmware/lookupsvc/lookupsvc.log`	Follow Lookup Service log
`tail -f /var/log/vmware/vsphere-ui/logs/vsphere_client_virgo.log`	Follow vSphere Client log
`tail -f /var/log/vmware/vpostgres/postgresql*.log`	Follow PostgreSQL log
`vc-support-bundle`	Generate full support bundle

VCHA

Command	Description
`/usr/lib/vmware-vcha/vcha-cli cluster-get-state`	VCHA cluster state

Networking

Command	Description
`ss -tlnp`	List all listening TCP ports
`ip addr show`	Show network interfaces
`ip route show`	Show routing table
`cat /etc/resolv.conf`	DNS configuration
`nslookup ${VC_FQDN}`	DNS resolution test

21. API Quick Reference

Authentication

Method	Endpoint	Description
POST	`/api/session`	Create session (Basic Auth) -- returns session token
GET	`/api/session`	Get current session info
DELETE	`/api/session`	Destroy session (logout)

Appliance Health

Method	Endpoint	Description
GET	`/api/appliance/health/system`	Overall system health
GET	`/api/appliance/health/mem`	Memory health
GET	`/api/appliance/health/storage`	Storage health
GET	`/api/appliance/health/database-storage`	Database storage health
GET	`/api/appliance/health/load`	CPU load health
GET	`/api/appliance/health/swap`	Swap health
GET	`/api/appliance/health/softwarepackages`	Software package health

Appliance Configuration

Method	Endpoint	Description
GET	`/api/appliance/ntp`	Get NTP servers
PUT	`/api/appliance/ntp`	Set NTP servers
GET	`/api/appliance/timesync`	Get time sync mode
PUT	`/api/appliance/timesync`	Set time sync mode (NTP/HOST)
GET	`/api/appliance/access/ssh`	Get SSH access status
PUT	`/api/appliance/access/ssh`	Enable/disable SSH
GET	`/api/appliance/networking`	Get network configuration

GET	`/api/appliance/networking/dns/servers`	Get DNS servers
GET	`/api/appliance/networking/dns/hostname`	Get hostname

Logging

Method	Endpoint	Description
GET	`/api/appliance/logging/forwarding`	Get syslog forwarding config
PUT	`/api/appliance/logging/forwarding`	Set syslog forwarding config
POST	`/api/appliance/logging/forwarding?action=test`	Test syslog forwarding

Inventory

Method	Endpoint	Description
GET	`/api/vcenter/datacenter`	List datacenters
GET	`/api/vcenter/cluster`	List clusters
GET	`/api/vcenter/host`	List hosts
GET	`/api/vcenter/vm`	List VMs
GET	`/api/vcenter/datastore`	List datastores
GET	`/api/vcenter/network`	List networks
GET	`/api/vcenter/folder`	List folders
GET	`/api/vcenter/resource-pool`	List resource pools

vCenter HA

Method	Endpoint	Description
GET	`/api/vcenter/vcha/cluster/mode`	Get VCHA mode
POST	`/api/vcenter/vcha/cluster?action=get`	Get full VCHA cluster status
POST	`/api/vcenter/vcha/cluster?action=failover`	Initiate VCHA failover

Certificates

Method	Endpoint	Description

GET	`/api/vcenter/certificate-management/vcenter/tls`	Get TLS certificate info
GET	`/api/vcenter/certificate-management/vcenter/trusted-root-chains`	List trusted root chains

Identity & SSO

Method	Endpoint	Description
GET	`/api/vcenter/identity/providers`	List identity providers

Common curl Pattern

```
# GET request pattern
curl -sk -X GET \
  "https://${VC_FQDN}/api/<endpoint>" \
  -H "vmware-api-session-id: ${VC_TOKEN}" | jq .

# POST request pattern
curl -sk -X POST \
  "https://${VC_FQDN}/api/<endpoint>" \
  -H "vmware-api-session-id: ${VC_TOKEN}" \
  -H "Content-Type: application/json" \
  -d '{ "key": "value" }' | jq .

# PUT request pattern
curl -sk -X PUT \
  "https://${VC_FQDN}/api/<endpoint>" \
  -H "vmware-api-session-id: ${VC_TOKEN}" \
  -H "Content-Type: application/json" \
  -d '{ "key": "value" }'
```

Session Management Best Practice

```
# Create session at start
export VC_TOKEN=$(curl -sk -X POST \
  "https://${VC_FQDN}/api/session" \
  -u "${VC_USER}:${VC_PASS}" | tr -d '"')

# ... run all health checks ...
```

```
# Destroy session at end
curl -sk -X DELETE \
  "https://${VC_FQDN}/api/session" \
  -H "vmware-api-session-id: ${VC_TOKEN}"
unset VC_TOKEN VC_PASS
```

Document Information:
Title: vCenter Server Health Check Handbook
Version: 1.0
Author: Virtual Control LLC
Date: March 2026
Classification: Internal Use
Platform: VMware Cloud Foundation 9.0 / vCenter Server 8.x
Disclaimer: Always test commands in a non-production environment first. Virtual Control LLC is not responsible for any issues arising from the use of commands in this document without proper testing.