SAP HANA

- SAP Production Incident – Troubleshooting SAP Router, APP Dispatcher, and HANA Dependency Failure

SAP Production Incident – Troubleshooting SAP Router, APP Dispatcher, and HANA Dependency Failure

Introduction

Recently I handled an SAP Production incident where users were suddenly unable to access critical SAP production services including SAP GUI

At first glance the issue looked like a network or firewall problem because users received a connection refused error when trying to connect to the SAP environment.

However, after tracing the entire SAP landscape step-by-step, the actual issue turned out to be a startup dependency problem involving:

- SAP Router
- SAP Application Dispatcher
- SAP HANA Database
- Web Dispatcher backend connectivity

This article shares the troubleshooting process from infrastructure level until application recovery.

> Note: Hostnames, IP addresses, SID names, and environment details have been anonymized for confidentiality.

Environment Overview

The SAP production landscape consisted of several separate virtual machines:

1. SAP Router VM
2. SAP Application VM
3. SAP Web Dispatcher VM
4. SAP Database VM

Architecture flow:

Client
↓
SAP Router
↓
SAP APP Dispatcher
↓
SAP HANA Database

Initial Symptoms

Users reported:

SAP GUI connection failed
web application inaccessible

SAP GUI error:

partner '<APP-IP>:3230' not reached
Connection refused

At this stage there were several possible causes:

- Firewall issue
- NAT issue
- SAP Router down
- Dispatcher down
- Database issue
- SAP service startup failure

Step 1 – Validate Network Connectivity

Initial external validation was performed using:

```
nmap -Pn -sT -sV \
-p 3299,3230,3330,50013,50014 <PUBLIC-IP>
```

Result:

```
3299 closed
3230 closed
50013 open
50014 open
```

Interesting finding:

```
SAPControl ports were alive
but SAP application ports were closed.
```

This usually means:

sapstartsrv running
but SAP application services not operational

Step 2 – SAP Router Investigation

SAP Router port:

3299

was confirmed not listening.

Further inspection on the SAP Router VM revealed that SAP Router was started manually using:

./saprouter -r -s 3299 ...

It was not configured as a proper persistent service.

Additional discovery:

The startup script previously used an incorrect parameter:

-S 3299

instead of:

-s 3299

After correcting and starting the service manually:

./saprouter -r -s 3299 ...

Result:

3299 LISTEN

SAP Router connectivity was restored successfully.

Step 3 – SAP APP Instance Investigation

Next, SAP instance status was checked:

sapcontrol -nr 30 -function GetProcessList
sapcontrol -nr 31 -function GetProcessList

Initial status:

Instance 31

msg_server GRAY
enq_server GRAY

Instance 30

disp+work GRAY
gateway stopped

Port validation:

3230 CLOSED

This explained the SAP GUI connection refusal.

Step 4 – Recover SAP Central Services

Instance 31 was started first:

sapcontrol -nr 31 -function StartSystem

Result:

msg_server GREEN
enq_server GREEN

However, instance 30 still failed to become healthy.

Step 5 – Dispatcher Failure Analysis

Dispatcher logs were analyzed:

cd /usr/sap/<SID>/D30/work

Important logs:

dev_disp
sapstart.log

Critical error found:

DpWpCheck: no more work processes

Multiple work processes terminated unexpectedly.

At this point the investigation shifted toward database dependency.

Step 6 – HANA Database Investigation

HANA status check:

HDB info

Result:

HANA services not fully running

Important HANA services such as:

- nameserver
- indexserver
- xsengine

were unavailable.

This explained why:

SAP APP work processes failed during startup.

The SAP dispatcher depended on HANA connectivity.

Step 7 – Start HANA Database

Database services were started manually:

HDB start

After startup:

HANA processes became active

Ports became available again.

Step 8 – Recover SAP APP Instance

After HANA became healthy:

sapcontrol -nr 30 -function StartSystem

Result:

disp+work GREEN
gwrd GREEN
icman GREEN

Port validation:

3230 LISTEN

SAP GUI connectivity was restored.

Step 9 – Verify Web Dispatcher

Web Dispatcher status:

sapcontrol -nr 00 -function GetProcessList

Initially:

YELLOW

Later:

GREEN

Backend connectivity recovered automatically after APP and HANA services became available.

Web application login page became accessible again.

Root Cause Analysis

The incident was ultimately caused by:

Several SAP services did not auto-start properly
after system startup/reboot.

Affected components:

- SAP Router service
- SAP HANA Database services
- SAP APP dispatcher dependency chain

Impact chain:

HANA DB not running
↓
SAP APP work processes failed
↓
Dispatcher shutdown
↓
Port 3230 unavailable
↓
SAP GUI connection refused
↓
Web Dispatcher backend degraded

Key Lessons Learned

1. Open SAPControl Port Does Not Mean SAP Is Healthy

Ports like:

50013
50014

may still be reachable even when SAP applications are completely down.

Because:

sapstartsrv can remain alive independently.

---

2. SAP Startup Dependency Order Matters

Correct startup sequence:

1. HANA Database
2. ASCS / Message Server
3. APP Dispatcher
4. Web Dispatcher
5. SAP Router

If APP starts before HANA:

work processes may terminate immediately.

---

3. SAP Router Should Use Proper Service Management

Running SAP Router manually from shell sessions is risky.

Better approach:

- systemd service
- persistent startup
- automatic recovery

---

4. Dispatcher Logs Are Extremely Important

The key clue came from:

dev_disp

which clearly showed:

all work processes terminated

without that log analysis, troubleshooting could easily remain focused on networking instead of backend dependencies.

Final Status

All SAP services recovered successfully:

[OK] SAP Router running
[OK] HANA running
[OK] SAP APP healthy
[OK] Dispatcher GREEN
[OK] Web Dispatcher GREEN
[OK] SAP GUI accessible
[OK] Web application accessible

Conclusion

This incident demonstrated how SAP outages can initially look like simple network issues while the real root cause actually exists deep in service dependency chains.

The troubleshooting flow that worked best was:

Network
→ SAP Router
→ SAP Instance
→ Dispatcher Logs
→ HANA Dependency
→ Service Recovery

In complex SAP environments, understanding the startup relationship between routing services, application dispatcher services, web dispatcher services, and backend databases is critical for fast recovery and accurate root cause analysis.