dB-electronics

Document Type: Lab Guide & Test Procedure
Target: PacketPilot Suite v1.0
Classification: Internal

PacketPilot Suite — Lab Guide & Test Procedure

Target hardware: Lenovo Legion (Windows + WSL2) + Cisco Catalyst 2960 | Last updated: 2026-05-06

PART 1 — LAB SETUP AND EXECUTION GUIDE

1.1 Prerequisites on Your Legion Laptop

Open **PowerShell as Administrator** and run each command once.

```
# Check Docker is installed and running
docker --version
docker compose version
# Expected: version numbers printed, no error

# If Docker Desktop is not running, start it
start "" "C:\Program Files\Docker\Docker\Docker Desktop.exe"

# Verify WSL2 (for Linux containers / CLI tools)
wsl --status
# Expected: "Default Distribution: Ubuntu" or similar
```

> **Note:** Docker Desktop on Windows runs its own Linux VM (WSL2 backend). All `docker` and `docker compose` commands in this guide are run from PowerShell (Windows), NOT inside WSL, unless explicitly shown.

1.2 Clone / Refresh the Suite Bundle

```
# Navigate to your workspace
cd C:\Users\Olore\workspace\packetpilot-suite

# Pull latest (if already cloned)
git pull origin master

# Verify the bundle contents
dir
# You should see: docker-compose.yml, Makefile, release.bat,
#                 README.md, keys/, docs/
```

1.3 Start the Full Suite

```
cd C:\Users\Olore\workspace\packetpilot-suite

# Option A — Using Docker Compose directly (recommended)
docker compose up -d

# Option B — Using make (Windows requires make from gnuwin32 or Chocolatey)
make start

# Wait 30 seconds for all services
Start-Sleep -Seconds 30
```

Screenshot checkpoint: `docker ps` should show 5 containers running:

Container	Image	Ports
`ppro-license`	`packetpilot-license-server`	`9000`
`ppro-analyze-api`	`packetpilot-analyze`	`8000`
`ppro-analyze-db`	`postgres:16-alpine`	`5432`
`ppro-ollama`	`ollama/ollama`	`11434`
`ppro-analyze-ui`	`packetpilot-analyze-ui`	`3000`

1.4 Verify All Services Are Up

Run each curl command in **PowerShell**:

```
# License Server
Invoke-RestMethod http://localhost:9000/ -ErrorAction SilentlyContinue
# Expected JSON: {"product":"PacketPilot License Server","version":"1.0.0","docs":"/docs"}

# Analyze API
Invoke-RestMethod http://localhost:8000/ -ErrorAction SilentlyContinue
# Expected JSON: {"product":"PacketPilot Analyze","version":"0.1.0","docs":"/docs"}

# Analyze API Health (includes license status)
Invoke-RestMethod http://localhost:8000/api/health -ContentType "application/json" | ConvertTo-Json
# Expected: status=ok, license block with valid/expired/grace flags

# Analyze UI (web interface)
Start-Process http://localhost:3000
# Expected: Login screen in your browser
```

1.5 Generate a License Key

```
$body = @{
    user     = "Lab Tester"
    expiry   = "2027-12-31"
    features = @{
        analyze_pro = $true
        config_pro  = $true
    }
} | ConvertTo-Json

$response = Invoke-RestMethod -Uri http://localhost:9000/licenses `
    -Method Post `
    -Body $body `
    -ContentType "application/json"

Write-Output "License key: $($response.key)"
# Expected output: PPRO-XXXXXX-YY  (copy this now — you'll use it in TC-002 and TC-003)
```

1.6 Apply License and Start Config App

```
# Set env vars for the Config app (run BEFORE launching)
$env:LICENSE_SERVER_URL = "http://localhost:9000"
$env:PPRO_LICENSE_KEY = "PPRO-XXXXXX-YY"   # <-- paste your key from Step 1.5

# Launch Config app
cd C:\Users\Olore\workspace\packetpilot-config
.\dist\PacketPilot Config.exe
# Or from source:
python main.py
```

1.7 Verify Analyze UI is Accessible

```
# Health check with license details
Invoke-RestMethod http://localhost:8000/api/health -ContentType "application/json" | ConvertTo-Json
# Expected license block:
# {
#   "valid": true,
#   "expired": false,
#   "in_offline_grace": false,
#   "expiry": "2027-12-31",
#   "days_left": 604,
#   "grace_days_remaining": 0,
#   "error": null
# }
```

PART 2 — CISCO 2960 LAB CONFIGURATION

> **Scope:** These commands are entered on a Cisco Catalyst 2960 connected to your network. Adjust IP addresses to match your lab environment.

2.1 Connect to the Switch

```
# From PowerShell or WSL
ssh admin@192.168.1.10
# or
ssh admin@192.168.1.10 -p 22
# Enter enable password when prompted
```

2.2 Basic Switch Identity

```
enable
configure terminal

! Hostname
hostname LAB-2960-01

! Domain name (required for crypto key)
ip domain-name lab.local

end
write memory
```

2.3 Generate RSA Crypto Key (Required for SSH)

```
configure terminal
crypto key generate rsa modulus 2048
end
write memory
```

2.4 Local User Account

```
configure terminal
username admin privilege 15 secret P@ssw0rd123
end
write memory
```

2.5 SSH Only on VTY Lines

```
configure terminal
line vty 0 4
 login local
 transport input ssh
 no exec-timeout
 no timeout login response
 exit
ip ssh version 2
ip ssh authentication-retries 3
ip ssh time-out 30
end
write memory
```

2.6 Management Interface (VLAN 1)

```
configure terminal
interface Vlan1
 ip address 192.168.1.10 255.255.255.0
 no shutdown
 exit
ip default-gateway 192.168.1.1
end
write memory
```

2.7 Verify SSH Access from Your Legion

```
# From PowerShell or WSL
ssh admin@192.168.1.10 -v
# Expected: connection opens, asks for password, drops into IOS shell

# Verify SSH version
ssh admin@192.168.1.10 -V
# Expected: OpenSSH_8.x or later (SSH-2.0)
```

2.8 Useful Verification Commands

```
show ip interface brief
show ssh
show users
show running-config | include username
show running-config | include line vty
```

PART 3 — FORMAL TEST CASES

TC-001 — Start Full Suite via Docker Compose

Test: Start full PacketPilot Suite with Docker Compose and verify all services respond

Pre-conditions: Docker Desktop running on Legion; `packetpilot-suite` directory present

Steps:

1. `cd C:\Users\Olore\workspace\packetpilot-suite`
2. `docker compose up -d`
3. Wait 45 seconds
4. `docker ps --format "table {{.Names}}\t{{.Status}}\t{{.Ports}}"`
5. `Invoke-RestMethod http://localhost:9000/`
6. `Invoke-RestMethod http://localhost:8000/api/health -ContentType "application/json"`
7. Open `http://localhost:3000` in browser

Expected: All 5 containers running; license server returns JSON; Analyze health shows `"status":"ok"` with license block; Analyze UI shows login screen

TC-002 — Hard License Gate in Config App

Test: Config app blocks startup without a valid license; shows license gate dialog

Pre-conditions: Suite running (TC-001 done); NO `PPRO_LICENSE_KEY` env var set

Steps:

1. Open PowerShell, `cd C:\Users\Olore\workspace\packetpilot-config`
2. Launch app WITHOUT license key: `python main.py`
3. Observe license gate dialog appears (title: "License Required")
4. Note the dialog has "Verify" and "Exit" buttons
5. Click "Exit" → app closes
6. Set key: `$env:PPRO_LICENSE_KEY = "PPRO-XXXXXX-YY"`
7. Launch again: `python main.py`
8. Observe: gate dialog does NOT appear; app proceeds to auth screen

Expected: Without key: gate dialog blocks all UI. With valid key: app starts normally. Status shows `"License: OK (expires YYYY-MM-DD)"`

TC-003 — Analyze Soft Gate (Valid vs. Missing License)

Test: Analyze API and UI remain fully usable with or without a license; only a warning banner appears

Pre-conditions: Suite running (TC-001 done)

Steps:	**Part A — With valid license:** 1. `Invoke-RestMethod http://localhost:8000/api/health -ContentType "application/json"` 2. Observe `"valid":true`, `"in_offline_grace":false` in license block 3. Open `http://localhost:3000` — no license warning banner **Part B — Without license:** 4. Stop Analyze API: `docker compose stop analyze-api` 5. Unset env var: `Remove-Item Env:\PPRO_LICENSE_KEY` 6. Restart: `docker compose up -d analyze-api` 7. Wait 10s, `Invoke-RestMethod http://localhost:8000/api/health` 8. Observe `"valid":false`, error message present 9. Open Analyze UI — warning banner visible at top: "License invalid or expired..."
Expected:	Both states: API accessible, UI usable. No HTTP errors or access denials. Only difference is a visible warning banner in Analyze UI

TC-004 — Add Cisco 2960 to Config and Run a Command

Test:	PacketPilot Config can add the Cisco 2960 and execute `show version` over SSH
Pre-conditions:	TC-001 and TC-002 complete; Cisco 2960 configured per Part 2; switch reachable at `192.168.1.10`
Steps:	1. In Config app: Dashboard → **Device Inventory** tab 2. Click **"Add Device"** 3. Fill in: Name= `LAB-2960-01`, IP= `192.168.1.10`, Type= `Switch`, Vendor= `Cisco` 4. SSH port: `22`, Auth method: `Local User` 5. Username= `admin`, Password= `P@ssw0rd123` 6. Click **"Save & Test"** — wait for success message 7. Go to **Config** tab, select `LAB-2960-01` 8. In command input: `show version` 9. Click **"Execute"** 10. Observe output in results pane (IOS version, uptime, serial)
Expected:	SSH connection succeeds; `show version` output appears in Config UI with no errors. Status bar shows `"License: OK"`.

TC-005 — Capture & Analyze: Send a Case from Config to Analyze

Test:	"Capture & Analyze" button in Config sends a new case to Analyze and opens the case in Analyze UI
Pre-conditions:	TC-004 complete (device added); Analyze API and UI running

Steps:

1. In Config app: **Config** tab
2. Select `LAB-2960-01` from device dropdown
3. Enter a CLI command, e.g. `show interfaces`
4. Click **"Capture & Analyze"** (⚡ button)
5. When prompted for a case title, enter: `Lab Test - Interface Status`
6. Click **"Send to Analyze"**
7. Note the case ID returned (e.g. `Case ID: abc-123`)
8. Open `http://localhost:3000` in browser
9. Find the case by title or ID
10. Open the case — verify CLI output is attached

Expected: Case appears in Analyze UI within ~10s of submission. CLI output is stored as ar artifact. Case status is "open".

TC-006 — Offline Grace Behavior

Test: When the license server is unreachable, Config and Analyze both enter grace mode correctly

Pre-conditions: Suite running (TC-001); valid license already verified in TC-002

Steps:

Part A — Config offline grace (< 30 days since last valid check):

1. Start Config with valid license → verify status bar: `"License: OK"`
2. Stop license server: `docker compose stop license-server`
3. Wait 5 seconds, re-launch Config
4. Observe: warning dialog: "License server unreachable. Running in offline grace mode."
5. Click OK → app continues, status bar: `"License: Grace mode"`

Part B — Config grace expired (> 30 days since last valid check):

7. Simulate expired cache: `docker compose down` , delete `data/` folder
8. Restart suite, generate new key, start Config, then stop license server
9. Edit `.license_cache.json` — set `last_checked` to > 30 days ago
10. Re-launch Config → observe: **hard block** (license gate dialog)

Part C — Analyze offline grace:

11. Start Analyze without a license key
12. Stop license server: `docker compose stop license-server`
13. Wait 10s, `Invoke-RestMethod http://localhost:8000/api/health`
14. Observe: `"in_offline_grace":true` if cache is fresh; `"valid":false` if stale

Expected: Config: graceful warning → app keeps running. Config with stale cache: hard block. Analyze: graceful banner, full access.

TC-007 — License Status in Health Endpoints

Test: License status is correctly reported in API health responses for both services

Pre-conditions: Suite running; valid license key active

Steps:	1. `Invoke-RestMethod http://localhost:8000/api/health -ContentType "application/json"` 2. Confirm the `license` object contains all fields: `valid`, `expired`, `in_offline_grace`, `expiry`, `days_left`, `grace_days_remaining`, `error`, `last_checked` 3. Check license server logs: `docker compose logs license-server --tail 2` 4. Look for `POST /validate 200` entries
Expected:	Health response contains full license block with no null errors. Server logs show successful validation requests.

TC-008 — Negative: Invalid License Key

Test:	Config correctly rejects a fake license key and blocks the app
Pre-conditions:	Suite running; Config app not running
Steps:	1. `$env:PPRO_LICENSE_KEY = "PPRO-FAKE1-99"` 2. `cd C:\Users\Olore\workspace\packetpilot-config; python main.py` 3. Observe license gate dialog: "License Required" with red title 4. Try entering `PPRO-FAKE1-99` in the gate dialog's input and click "Verify" 5. Observe: status label turns red: "× Invalid: License key not found" 6. Click "Exit" → app closes cleanly 7. Verify with Analyze API: `Invoke-RestMethod -Method Post -Uri http://localhost:9000/validate -Body '{"key":"PPRO-FAKE1-99","hw_id":"test"}' -ContentType "application/json"`
Expected:	Gate dialog shows specific error: "License key not found". API returns `"valid":false`, `"error":"License key not found"`. App does not reach au screen.

TC-009 — Negative: Unreachable Switch (SSH Failure)

Test:	Config correctly handles a switch that is unreachable over SSH
Pre-conditions:	Config app running; switch powered off or IP unreachable
Steps:	1. Config app → **Device Inventory** tab 2. Add a device with a fake IP: `10.255.255.1`, SSH port `22` 3. Click **"Save & Test"** 4. Observe error message: "Connection failed: Network is unreachable" or similar 5. Go to **Config** tab, select the unreachable device 6. Enter `show version`, click **"Execute"** 7. Observe error in results pane: SSH timeout or connection refused
Expected:	Specific, actionable error message. No crash. App remains stable.

APPENDIX A — Quick Reference

Service URLs

Service	URL	Notes
License Server API	`http://localhost:9000`	Admin UI: `http://localhost:9000/docs`
Analyze API	`http://localhost:8000`	Docs: `http://localhost:8000/docs`
Analyze Health	`http://localhost:8000/api/health`	Includes license status
Analyze UI	`http://localhost:3000`	Browser access
Postgres	`localhost:5432`	Internal only

Generate a License Key (API)

```
Invoke-RestMethod -Method Post -Uri http://localhost:9000/licenses `
  -Body (@{
    user="Test User"; expiry="2027-12-31";
    features=@{analyze_pro=$true; config_pro=$true}
  } | ConvertTo-Json) `
  -ContentType "application/json"
```

Container Management

```
# Stop all services
docker compose down

# Restart all services
docker compose up -d

# View logs for a specific container
docker compose logs -f license-server
docker compose logs -f analyze-api
docker compose logs -f analyze-ui

# Restart just the license server (no downtime to other services)
docker compose restart license-server
```

Clear All Data (Fresh Start)

```
docker compose down -v
Remove-Item -Recurse -Force C:\Users\Olore\workspace\packetpilot-suite\data -ErrorAction SilentlyContinue
# Then re-run: docker compose up -d
```

APPENDIX B — Cisco 2960 Command Cheat Sheet

```
! Save configuration
write memory
copy running-config startup-config

! View interface status
show ip interface brief
show interfaces status

! View MAC address table
show mac address-table

! View VLANs
show vlan brief

! View SSH connections
show ssh

! View logged-in users
show users

! Backup config (from Config app terminal)
copy running-config tftp:
! Then enter TFTP server IP when prompted
```

APPENDIX C — Troubleshooting

Symptom	Cause	Fix
`docker compose up -d` fails	Port already in use	`netstat -ano | findstr "9000 8000 3000 5432 11434"` and stop conflicting service
License server returns 404	Wrong image tag	`docker compose pull` then `docker compose up -d`
Config app shows "License: Grace mode"	License server stopped	`docker compose up -d license-server`

Analyze UI blank page	Backend not ready	Wait 30s after `docker compose up -d`, or `docker compose restart analyze-ui`
SSH to switch fails	VTY not configured for SSH	Ensure `transport input ssh` under `line vty 0 4`
"Capture & Analyze" button missing	Not on Config tab	Navigate to Config → Config tab, select a device first
`invoke-restmethod` timeout	Container not responding	`docker ps` to verify container is Up; `docker compose logs <svc>` to check errors