AI-04739 Homeassistant vs Tailscale Fix

SunPower PVS5 + Enhanced SunPower Logs — Analysis

Rigel, these logs are useful. They show that your Home Assistant OS installation, Enhanced SunPower integration, and local PVS5 communication are functioning successfully. However, the data also reveals some inconsistencies in the power readings and timestamps that deserve further investigation—especially considering your earlier report of three solar panels not producing.

I'll break this down into the system health, solar inverter readings, power meter data, and what we can and cannot conclude about the suspected dead panels.

1. Executive Summary

Home Assistant / Integration

Operational

The integration successfully polled the PVS5 and returned device data.

Local PVS5 communication

100% success

166 successful polls, zero failed polls, and approximately 0.97 seconds average response time.

23 inverters

Reported working

All 23 devices report a working state in the provided data. This does not prove that every panel is producing normal power.

Power meter interpretation

Investigate

The power meter values contain apparent inconsistencies, including very low power factor on one meter and differing timestamps.

Primary conclusion: Your communication infrastructure appears healthy. The logs do not establish whether the three suspected panels are defective, disconnected, shaded, or otherwise underperforming.

2. Home Assistant and Integration Health

Home Assistant OS

Running

Virtual Machine

- Installation: Home Assistant OS
- Version: `2026.9.3`
- OS version: `18.3`

- Architecture: `x86_64 / amd64`
- Docker: `29.7.2`
- Python: `3.14.6`

Enhanced SunPower

Installed

Local polling

- Integration version: `2026.9.3`
- HACS version: `2.0.5`
- Integration type: Device
- Config flow: Enabled
- Requirements: None

What this tells us

The Enhanced SunPower integration is installed and recognized by Home Assistant. The integration manifest identifies it as a local polling integration, which aligns with your goal of monitoring the PVS5 locally rather than depending on the former SunPower cloud service.

The setup completed in approximately 1.25 seconds for the configured entry, and the coordinator is reporting successful updates.

3. Polling and Communication Diagnostics

The most important part of the diagnostic data is here:

|
Metric
|
Value
|
| --- | --- |
|
Total polls
|
166
|
|
Successful polls
|
166
|
|

Failed polls

|

0

|

|

Consecutive failures

|

0

|

|

Success rate

|

100%

|

|

Average response time

|

0.965 seconds

|

|

Polling interval

|

300 seconds (5 minutes)

|

|

Last success

|

2026-09-21 05:04:05

|

|

Integration uptime

|

~13.8 hours

|

Communication assessment

Based on the supplied diagnostic snapshot, Home Assistant has maintained successful communication with the PVS5 during the recorded polling period.

What this rules out

The data makes the following explanations less likely for the period covered by the logs:

- A continuously broken connection between Home Assistant and the PVS5.
- A polling failure preventing the integration from retrieving device data.
- A currently persistent authentication failure in this integration's configured connection.

It does not rule out intermittent network problems outside the recorded period, faulty equipment, or issues affecting the PVS5's own communication with individual inverters.

4. PVS5 Controller Health

The PVS5 controller reports the following:

|
Parameter

|

Value

|

Interpretation

|
| --- | --- | --- |
|

Model

|

PV Supervisor PVS5

|

Your local SunPower controller

|
|

Hardware

|

PVS5 3.3

|

Hardware revision

|
|

Software

|

0.0.25.5412

|

Installed software version

|
|
State
|
Working
|
Controller-reported state
|
|
Uptime
|
272,709 seconds
|
Approximately 3.16 days
|
|
CPU load
|
0.57
|
Low reported CPU load
|
|
RAM usage
|
80%
|
Worth monitoring
|
|
Flash usage
|
2%
|
Low reported usage
|
|

Error count

|

0

|

No reported errors in this field

|
|

Communication errors

|

0

|

No reported errors in this field

|
|

Skipped scans

|

0

|

No skipped scans reported

|

Important observation: RAM usage is 80%

The PVS5 reports `ram_usage_percent: 80` .

That is not automatically a fault. However, because you previously experienced PVS5-related monitoring problems and are interested in long-term reliability, it is a useful metric to monitor.

The log also reports:

- `dl_mem_used: "0"`
- `dl_flash_avail: "0"`
- `flashwear_percent: null`

These fields should not be treated as proof of zero memory consumption, zero available flash, or unknown flash failure. The integration's diagnostic representation may not populate these fields meaningfully. The RAM percentage is the clearer reported value, but we would need the controller's own monitoring definitions or historical readings to assess whether it is abnormal.

Suggested monitoring

Track RAM usage over several days and compare it with:

- PVS5 uptime.
- Controller reboots.
- Polling failures.

- Any increasing error counts.

A single 80% reading is not enough to diagnose a memory leak or failing controller.

5. Inverter Data: All 23 Report "Working"

The integration reports:

Inverter inventory

23 devices

23 of 23 reported state: Working

Device state reported by the PVS5 snapshot, not an independent confirmation of electrical output.

Every inverter in the provided raw data has:

- `DEVICE_TYPE: Inverter`
- `MODEL: AC_Module_Type_D`
- `STATE: working`
- `STATEDESCR: Working`

The system reports 23 inverters, which corresponds to your reported 23-panel installation if each panel has its own AC microinverter.

Critical distinction: Working does not equal producing normally

This is especially important for your suspected three dead panels.

A microinverter can report a working state even if the associated panel is producing little or no power due to:

- Shading or obstruction.
- A disconnected or damaged DC connection.
- Panel-side electrical issues.
- A communication state that does not fully reflect real-time production.
- Low-light conditions.
- A fault not captured in the specific fields exposed by the integration.

The log alone cannot identify which three panels are underperforming. The inverter serial numbers and panel mapping from Freedom Power would be needed to correlate device-level output to physical roof locations.

6. The Most Important Data Issue: Timestamps

Your logs contain different timestamps. We need to be careful not to compare them as though they all represent the same instant.

PVS5 controller

Controller data

Reported

`2026-09-21 05:04:05`

`DATATIME: 2026,09,21,10,04,04`

Your Home Assistant timezone is `America/Chicago`, and on September 21, 2026, this corresponds to a five-hour difference between the local CDT clock and UTC.

The controller's `DATATIME` of 10:04:04 is consistent with UTC while the integration's `last_success_time` is 05:04:05 local time. This suggests the timestamps may be using different time bases.

Inverter data

The inverter records show:

`DATATIME: 2026,09,21,00,15,00`

The provided inverter entries all show this same timestamp, while the PVS5 controller and integration diagnostic timestamps are around 05:04 local / 10:04 UTC.

This creates an important question:

> Are the inverter readings current at the time of the poll, or are they the most recently recorded readings from an earlier timestamp?

The logs alone do not resolve this. The integration may be returning cached or separately timestamped device measurements. The timestamp format and time zone should be verified before using these readings for a real-time production diagnosis.

7. Inverter Production: Why the Readings Are So Low

Let's examine the inverter readings.

Example:

|
Field

|
Example value

|
| --- | --- |
|
State

|
Working

|
|
`p_3phsum_kw`

|
0.000655 kW

|
|
`p_mppt1_kw`

|

0.00211 kW

|

|

v_mppt1_v

|

50.235 V

|

|

i_mppt1_a

|

0.02 A

|

|

Frequency

|

59.995 Hz

|

|

Temperature

|

40°C

|

The `p_3phsum_kw` value of `0.000655 kW` equals 0.655 watts.

That is extremely low output for a solar microinverter under meaningful daylight production, but it may be perfectly consistent with nighttime or near-zero-light conditions. The inverter timestamp in this data is `00:15`, and we cannot establish from the raw log whether it is local time, UTC, or another clock.

What the readings indicate

The inverters show:

- Grid voltage around 246 V.
- Frequency around 60 Hz.
- Temperatures in the mid-30s to mid-40s °C.
- Very low active power readings in the provided snapshot.

The voltage and frequency values are not, by themselves, evidence of a grid abnormality. The very low power values must be interpreted in conjunction with the timestamp and actual daylight conditions.

Do not use these low readings alone as proof that the solar panels are dead.

8. Power Meter Analysis

The power meter data is the most interesting part of this log because it gives us a view of the overall electrical system.

There are three power meters:

|
Meter

|

Model

|

Reported state

|
| --- | --- | --- |
|

1

|

PVS5M0400p

|

Working

|
|

2

|

PVS5M0400c

|

Working

|
|

3

|

PVS5M0400c

|

Working

|

Only two power meter entries are visible in the supplied raw PVS data, while the device summary reports three power meters. That is a detail worth checking in the integration's entity list or full raw response.

Meter 1: PVS5M0400p

Power meter P

Working

0.0173

kW

Reported active power

Power factor

0.023924

Reactive power

−0.71849 kvar

Apparent power

0.731251 kVA

Timestamp

09:55

The combination of low active power, significant reactive power relative to active power, and a power factor near zero deserves investigation. It could reflect the electrical characteristics or operating condition of that meter's circuit, but we cannot identify the cause from this snapshot alone.

Meter 2: PVS5M0400c

Power meter C

Working

0.844439

kW

Reported active power

Power factor

0.561414

Reactive power

0.797229 kvar

Apparent power

1.369899 kVA

Phase 1 power

0.612335 kW

Phase 2 power

0.232104 kW

Timestamp

09:55

This meter shows substantially higher active power than meter P.

Its phase-level values add up to approximately:

0.612335+0.232104=0.844439 kW0.612335 + 0.232104 = 0.844439\text{ kW}0.612335+0.232104=0.844439 kW

That matches the reported total active power.

Important limitation

We don't yet know the exact electrical wiring arrangement represented by meter P and meter C. The model names and readings alone do not establish which meter is measuring solar production, grid import/export, or household consumption.

Therefore, we should not label the meter readings as import or export without verifying the CT installation and integration definitions.

9. Potential Issue: CT Scaling and Meter Measurements

The power meter data includes current transformer (CT) scale factors:

|
Meter

|

CT scale factor

|
| --- | --- |
|

P meter

|

50

|
|

C meter

|

100

|

The C meter reports:

- `p1_kw: 0.612335`
- `p2_kw: 0.232104`
- `i1_a: 6.86047`
- `i2_a: 4.084581`
- `v1n_v: 125.109917`
- `v2n_v: 125.400673`

The presence of CT scale factors is important because incorrect CT orientation, scaling, phase association, or configuration can affect the meaning of power readings.

What we can infer from the provided values

The two phase power readings sum to the total reported active power. The phase voltages are around 125 V, which is consistent with a nominal 120/240 V split-phase system.

However, the log does not tell us:

- Whether the CTs are installed in the correct orientation.
- Which CT is connected to which phase.
- Whether the meter's configured CT scale matches the physical CT.
- Whether the integration is displaying import/export with a signed convention.
- Whether the meters are configured for the intended measurement locations.

These are the items to verify before diagnosing an electrical or solar production fault based on the meter readings.

10. What Does This Say About Your Three Dead Panels?

You previously mentioned that three panels were not working and that Freedom Power has the panel mapping.

This diagnostic snapshot provides a starting point but does not prove the fault location.

Fault assessment

Confirmed from the logs

- PVS5 is accessible.
- The Enhanced SunPower integration is installed.
- All 23 inverter devices report `Working`.
- Polling is currently successful.

Not established

- Which three panels are underperforming.
- Whether any specific microinverter is defective.
- Whether the issue is caused by a loose MC4 connector, shading, panel damage, or another electrical condition.
- Whether the current low inverter power values are nighttime or daylight measurements.

The missing piece: Device-level comparison during daylight

For your fault investigation, the most valuable data would be a simultaneous daytime snapshot of all 23 inverters with:

- Device serial number.
- Timestamp.
- Active power (`p_3phsum_kw`).
- MPPT voltage.
- MPPT current.
- Inverter state.
- Any available error or fault fields.

Then we can compare inverter output across the array and correlate the suspect inverter serial numbers with the Freedom Power panel map.

11. Diagnostic Architecture: What Is Working?

Here's a simplified interpretation of your monitoring stack.

Solar array

23 panels / microinverters

Physical system

Local data collection

SunPower PVS5

Local controller and device data

Working

Local polling over network

Home Assistant OS

VM on your lab infrastructure

Running

Enhanced SunPower integration

Solar monitoring and diagnostics

Device state, energy readings, power meter data

Data available

The monitoring path is operational in this snapshot. The remaining challenge is to validate the accuracy and timing of the electrical measurements, not simply to establish network connectivity.

12. Recommended Next Steps

I would investigate this in the following order.

1.

Capture a daytime snapshot

Collect all 23 inverters during strong sunlight, preferably around solar noon or during a period of stable irradiance.

Goal: Determine whether any inverter consistently reports negligible power compared with the others.

Validate timestamp interpretation

Compare the PVS5's `DATATIME` and the integration's polling time with the actual time on the controller.

Goal: Make sure the readings represent the period you think they do.

Export the inverter data into a table

Include serial number, power, MPPT voltage, current, temperature, and state.

Goal: Create a 23-inverter comparison and identify outliers.

Obtain the Freedom Power panel map

Match the physical panel positions to the microinverter serial numbers.

Goal: Identify where the low-output devices are physically located.

Investigate the suspect panels

Once the inverter-level evidence points to specific devices, inspect the relevant panel, connectors, and wiring using safe procedures.

Goal: Distinguish panel-side issues from microinverter, wiring, or monitoring issues.

13. Bottom Line

Your logs show a healthy monitoring connection but not a confirmed healthy solar array.

|
Question

|

Current conclusion

|
| --- | --- |
|

Is Home Assistant OS running?

|

Yes

|
|

Is Enhanced SunPower installed?

|

Yes

|
|

Is the PVS5 being polled successfully?

|

Yes

|
|

Are all 23 inverters reported as working?

|

Yes

|
|

Are all 23 panels proven to be producing normally?

|

No

|
|

Can we identify the three suspect panels from this snapshot?

|

No

|

|

Is the power meter data worth further investigation?

|

Yes

|

|

Is there enough evidence to blame loose MC4 connectors?

|

No

|

The next useful step is to collect a daylight inverter snapshot with serial numbers and output, then compare it against the panel map. That will turn the investigation from a general suspicion into a device-level fault analysis.

What I would like to see next

If you can provide a daytime log from the Enhanced SunPower integration, preferably including the full 23-inverter readings, I can help build a structured comparison with:

- Output power ranking and outlier detection.
- MPPT voltage/current analysis.
- Potentially abnormal device identification.
- A fault tree for the three suspected non-producing panels.
- A panel-to-inverter diagnostic worksheet for Freedom Power.

This current log is a good foundation for that work.

*Here's the log to HA Tailscale. I'm trying to understand why I can't reach my HA Tailscale using **Http://100.102.103.79:8123**. I now able to connect to my hyper-v (TX) using my iphone hotspot on my laptop)*

Tailscale → Home Assistant (HAOS) Connectivity Diagnosis

Rigel, this log is much more revealing than the earlier solar monitoring log. Tailscale is starting successfully, obtaining the expected IP address, and configuring forwarding toward your Home Assistant host. That means we should focus on the connection between Tailscale's forwarding layer and Home Assistant's web service, rather than immediately assuming the Tailscale account or VPN is broken.

Your current setup:

iPhone hotspot

Internet connection for your laptop

Laptop network connection

Windows laptop

Can now connect to your TX Hyper-V environment

Reachable

Hyper-V networking

Home Assistant OS VM

TX lab · Local IP reported by log: 192.168.1.159

Tailscale add-on forwarding

Tailscale address

100.102.103.79

Assigned

Target web service

Home Assistant Web UI

Target: port 8123

Not yet verified

The important distinction is that a working Tailscale connection does not automatically prove that Home Assistant's HTTP service is reachable through that connection.

1. What the logs prove

A. Tailscale is running and assigned the correct IP

The log reports:

```
Switching ipn state Starting -> Running (WantRunning=true, nm=true)
health(warnable=warming-up): ok
```

And:

```
peerapi: serving on http://100.102.103.79:53217
```

The Tailscale add-on successfully brings up its networking stack and starts its peer API. The address `100.102.103.79` is the expected Tailscale IPv4 address shown in your log.

a0d7b954_tailscale_2026-09-21T10-13-11.272Z.log

Important: Port `53217` is the Tailscale peer API port, not Home Assistant's web UI port `8123`. The fact that Tailscale is listening on its own port does not confirm that HA is listening on port 8123.

B. The Tailscale add-on is configured to forward traffic

This is the most relevant line:

```
Forwarding incoming tailnet connections directed to 100.102.103.79
to the host's 192.168.1.159 address (IPv4)
```

The log also reports IPv6 forwarding to the host.

a0d7b954_tailscale_2026-09-21T10-13-11.272Z.log

This means the add-on is attempting to forward incoming tailnet connections to the host's LAN address.

That is encouraging, but it leaves an essential question:

> Is Home Assistant actually listening on `192.168.1.159:8123`, and is the forwarding path reaching that service?

We need to test that directly.

2. Potential problem: You are testing the wrong layer

Your current URL is:

```
http://100.102.103.79:8123
```

This is the correct general format for accessing an HTTP service over Tailscale, assuming the service is listening on port 8123 and the forwarding setup is correct.

But there are three separate network paths to distinguish:

Path 1 — Laptop → Hyper-V host

You said this now works through the iPhone hotspot. This confirms some level of connectivity to the TX environment, depending on how you're connecting.

Path 2 — Laptop → Tailscale IP

We need to test whether your laptop can reach `100.102.103.79` on port 8123, rather than merely seeing the device in the Tailscale admin console.

Path 3 — Tailscale forwarding → HA web service

The log shows forwarding toward `192.168.1.159`. We need to establish whether port 8123 is open and serving Home Assistant at the destination.

3. Let's diagnose systematically

Since you can now access your Hyper-V host using your laptop's iPhone hotspot connection, start with the Windows PowerShell checks. We can determine the failure point without changing any network settings yet.

Step 1 — Test the Tailscale IP from your laptop

Open PowerShell on your Windows laptop.

Run:

PowerShell

```
Test-NetConnection 100.102.103.79 -Port 8123
```

Then run:

PowerShell

```
Test-NetConnection 100.102.103.79 -Port 53217
```

How to interpret the results

|

Test

|

Interpretation

|

| --- | --- |

|

Port 8123 succeeds

|

Something is accepting TCP connections on HA's target port.

|

|

Port 8123 fails

|

Could be a forwarding issue, firewall, listener issue, or unreachable Tailscale path.

|

|

Port 53217 succeeds

|

Tailscale peer API is reachable, but this does not prove HA web access.

|

|

Both fail

|

Investigate the Tailscale path and network reachability.

|

Do not treat a failed 53217 test alone as a failure of Tailscale. The peer API is not the same service as Home Assistant.

Step 2 — Test the Home Assistant host locally

You need to determine whether the Home Assistant web service is reachable at the LAN IP shown in the logs.

From a machine that can reach the HAOS host's LAN, test:

PowerShell

```
Test-NetConnection 192.168.1.159 -Port 8123
```

If you can run a browser from a machine on the same LAN:

```
http://192.168.1.159:8123
```

Interpretation

If LAN access succeeds but Tailscale access fails: Focus on Tailscale forwarding, firewall rules, or the destination address.

If LAN access fails too: Focus on Home Assistant's web service, VM networking, the IP address, or firewall rules before investigating Tailscale.

4. Verify the IP address on the actual HAOS system

Your log contains:

```
eth0: 192.168.1.159/24
```

This is the IP address Tailscale is forwarding toward in the supplied log.

a0d7b954_tailscale_2026-09-21T10-13-11.272Z.log

However, we should verify that the address is still assigned to the Home Assistant host and that the current network arrangement hasn't changed after your Hyper-V and hotspot work.

From the Home Assistant Terminal / SSH environment, run:

Bash

```
ha network info
```

If you have access to the host shell:

Bash

```
ip addr
```

Also check:

Bash

```
ip route
```

We want to confirm:

- The active LAN IP.
- The default gateway.
- The interface carrying the `192.168.1.0/24` network.
- Whether the IP changed after the network configuration changes.

Don't assume that an IP address appearing in yesterday's Tailscale log is necessarily the current IP.

5. Check whether Home Assistant is listening on port 8123

The Tailscale forwarding line shows the target host address but does not demonstrate that the HTTP service is accepting connections.

If you have access to the HAOS host's shell, check:

Bash

```
ss -lntp | grep 8123
```

Depending on the Home Assistant OS shell environment, `ss` may not be available or may not expose all relevant processes.

You can also test the local service with:

Bash

```
curl -I http://127.0.0.1:8123
```

And:

Bash

```
curl -I http://192.168.1.159:8123
```

Expected outcome

A functioning Home Assistant web server will typically return an HTTP response, such as:

```
HTTP/1.1 200 OK
```

or an authentication-related response such as:

```
HTTP/1.1 401 Unauthorized
```

An HTTP response of `401`, `302`, or `200` still demonstrates that you reached an HTTP server. A connection timeout or refusal points to a different type of problem.

The exact response may vary with Home Assistant's configuration and version.

6. One possible issue: Hyper-V external switch and hotspot routing

You mentioned:

> I now able to connect to my hyper-v (TX) using my iphone hotspot on my laptop.

This is important context.

Your Hyper-V environment has an external virtual switch, and the laptop's active network connection is now an iPhone hotspot. If your Hyper-V host's external switch is bound to a particular physical adapter, changing the laptop's active internet connection can alter the path to the VM.

However, I cannot conclude from the Tailscale log that Hyper-V's virtual switch is the cause. The log indicates that Tailscale's own interface was created and that forwarding was configured.

We need to know exactly how you are connecting:

- Is the Hyper-V host directly connected to the iPhone hotspot?
- Is the laptop acting as a bridge or router?
- Is the Hyper-V host connected through a physical Ethernet adapter?

- Is the HAOS VM on an external switch that reaches the same LAN as the Tailscale forwarding target?

These details affect whether the target `192.168.1.159` remains reachable from the forwarding layer.

7. Another possible issue: The Tailscale add-on's forwarding design

Your logs show that Tailscale is configured to forward incoming connections directed to its Tailscale IP toward the host's LAN IP.

The relevant lines:

```
Forwarding incoming tailnet connections directed to
100.102.103.79 to the host's 192.168.1.159 address
```

This is an important distinction from a traditional Tailscale installation on a normal Linux server.

In a standard Linux installation, Tailscale usually creates a network interface on the host and routes traffic to services on that host. In a Home Assistant add-on environment, the add-on may be running in its own container/network context and forwarding traffic to the HAOS host.

That forwarding mechanism can work, but it creates an additional layer to validate:

```
Tailscale IP → Add-on forwarding → HAOS host IP → HA service :8123
```

If the forwarding target is incorrect or the service isn't available at the target address, you can see a healthy Tailscale status while the browser still fails.

8. The error message will tell us where to focus

When you enter:

```
http://100.102.103.79:8123
```

What exactly happens in Edge?

Select the browser error

ERR_CONNECTION_TIMED_OUT

ERR_CONNECTION_REFUSED

ERR_CONNECTION_RESET

ERR_ADDRESS_UNREACHABLE / Network unreachable

Blank page / keeps loading

Home Assistant login page appears

Other / different error

9. My recommended next action

Let's not change the Tailscale configuration or rebuild your Hyper-V network yet.

Run these two commands from your laptop and share the output:

PowerShell

```
Test-NetConnection 100.102.103.79 -Port 8123
```

PowerShell

```
Test-NetConnection 100.102.103.79 -Port 53217
```

Then tell me whether you can reach Home Assistant locally through:

```
http://192.168.1.159:8123
```

If you cannot access that address from the laptop, let me know how the laptop is connected to the Hyper-V host.

What we are trying to establish

Diagnostic objective

Identify whether the failure is occurring at:

1. Tailscale connectivity.
2. Tailscale add-on forwarding.
3. Home Assistant host IP or network path.
4. Home Assistant HTTP service on port 8123.

The log suggests that the Tailscale component is operational. The next test should tell us whether the service at port 8123 is reachable through the assigned Tailscale IP.

Tailscale Analysis — `homeassistant-tx`

This information confirms that Home Assistant is connected to your Tailscale network and appears reachable at the Tailscale layer. However, it does not yet prove that Home Assistant's web interface is accessible on TCP port `8123`.

1. What your Tailscale information confirms

Tailscale node is connected

`homeassistant-tx` is shown as Connected, with the expected Tailscale IPv4 address:

Tailscale IPv4

100.102.103.79

Operating system

Home Assistant OS / Linux

Tailscale version

1.102.4

Preferred relay

Dallas — 7.10 ms

UDP connectivity

Yes

Direct connection

Varies: No

The Dallas relay latency of 7.10 ms is encouraging, and UDP is available. The fact that Tailscale uses a relay rather than a direct peer-to-peer connection is not, by itself, a problem.

Your earlier Tailscale log also showed that Tailscale was running and forwarding traffic addressed to `100.102.103.79` toward the host's local address `192.168.1.159`. That supports the conclusion that the Tailscale forwarding component is configured, but it does not verify that Home Assistant is listening on port `8123`.

2. Important distinction: endpoints versus Home Assistant port

Your endpoint list contains:

```
192.168.1.159:52228
172.30.32.1:52228
172.30.232.1:52228
136.62.46.118:5744
```

These are Tailscale network endpoints, not proof that Home Assistant's web service is running on port `8123`.

The connection you want is:

```
Laptop
   |
   | Tailscale
   ▼
100.102.103.79:8123
   |
   | Tailscale forwarding / local network
   ▼
192.168.1.159:8123
   |
   ▼
Home Assistant web interface
```

The key unknown is whether the final service is available at:

```
192.168.1.159:8123
```

3. Run these tests from your Windows laptop

Open PowerShell on the laptop that is connected through your iPhone hotspot.

Test A — Tailscale IP and Home Assistant port

PowerShell

```
Test-NetConnection 100.102.103.79 -Port 8123
```

Test B — Tailscale peer API

PowerShell

```
Test-NetConnection 100.102.103.79 -Port 53217
```

The second test is diagnostic only. Port `53217` was associated with Tailscale's peer API in your earlier log; it is not the Home Assistant interface.

Test C — Local LAN address

If your laptop can access the `192.168.1.x` network, run:

PowerShell

```
Test-NetConnection 192.168.1.159 -Port 8123
```

You can also try:

```
http://192.168.1.159:8123
```

Do not expect Test C to work through the iPhone hotspot unless the laptop has a route to the TX LAN. The most important test from your current remote connection is Test A.

4. Interpret the result

|
Result
|
Likely meaning
|
| --- | --- |
|
`TcpTestSucceeded: True` for `100.102.103.79:8123`
|
Network path and TCP service are reachable; investigate browser URL, HTTP/HTTPS, or Home Assistant response
|
|
`TcpTestSucceeded: False` for `100.102.103.79:8123`
|
Port 8123 is blocked, not forwarded, or Home Assistant is not listening

|

|

Port `53217` succeeds but `8123` fails

|

Tailscale is reachable, but Home Assistant's web service path remains unresolved

|

|

Both ports fail

|

Could indicate a Tailscale access, firewall, forwarding, or routing issue

|

|

`192.168.1.159:8123` succeeds locally but Tailscale IP fails

|

Home Assistant works on the LAN, but Tailscale forwarding or access configuration is the likely focus

|

|

Both `192.168.1.159:8123` and `100.102.103.79:8123` fail

|

Investigate Home Assistant's listener, IP address, or service availability first

|

5. Check Home Assistant directly

Open the Home Assistant terminal or SSH session and run:

Bash

```
ss -lntp | grep 8123
```

Then:

Bash

```
curl -I http://127.0.0.1:8123
```

And:

Bash

```
curl -I http://192.168.1.159:8123
```

The expected outcome is an HTTP response such as:

```
HTTP/1.1 200 OK
```

or potentially:

```
HTTP/1.1 401 Unauthorized
```

A `401` would still indicate that the web service is responding. A connection refusal or timeout would require further investigation.

6. TLS certificate status

Your Tailscale page says:

```
Domain:
homeassistant-tx.tail3f76f1.ts.net

Status:
No certificate found
```

This is not necessarily the cause of the current problem.

You are currently trying to use:

```
http://100.102.103.79:8123
```

That is plain HTTP over the Tailscale network. A Tailscale HTTPS certificate is only relevant if you want to access Home Assistant using a hostname such as:

```
https://homeassistant-tx.tail3f76f1.ts.net
```

For now, continue testing with:

```
http://100.102.103.79:8123
```

My current assessment

|

Component

|

Status

|

| --- | --- |

|

Home Assistant Tailscale node

|

Connected

|

|

Tailscale IPv4 address

|

Correctly identified

|

|

Tailscale UDP

|

Available

|

|

Dallas relay

|

Available, 7.10 ms

|

|

Tailscale service

|

Previously shown running in the log

|

|

Tailscale forwarding

|

Previously shown configured

|

|

Home Assistant TCP port `8123`

|

Not yet verified

|

|

Home Assistant web listener

|

Not yet verified

|

Next step: Send me the complete output of Test A:

PowerShell

```
Test-NetConnection 100.102.103.79 -Port 8123
```

That result will help us determine whether to focus on the Home Assistant service itself or the Tailscale forwarding path.

Remember when I switched from spectrum to gFiber a few days ago, I thought I was smarter than you and did not follow verbatim your instructions.

Here's what I did:
I turned off my WLAN-0
I named my gFiber ssid as WLAN-0
I power cycle my PVS5 controller
I then got a different ip of the pvs5 but I don't remember what was the original ip.
I did scan now and found the sunpower ip is now .185

Pilot error??

Yes, Rigel—this was probably a small case of "pilot error," but not a disastrous one. 😄 More accurately, you performed a reasonable network migration, but the consequences of changing the SSID and restarting the PVS5 included a new DHCP address that we needed to account for.

The good news: your scan indicates that the SunPower device is back on the `192.168.1.x` network.

1. What your scan shows

Your network scanner identifies:

SunPower device

Detected

IP address

192.168.1.185

Vendor

SunPower Corp.

MAC shown in scan

00:22:f2:04:98:ab

Network

192.168.1.0/24

This is consistent with your Google Fiber LAN and suggests that the PVS5 successfully reconnected to the network.

2. What actually happened

Your sequence was:

1. Turned off the old WLAN-0
   You disabled the previous Spectrum wireless network.
2. Renamed the Google Fiber SSID to WLAN-0
   This allowed the PVS5 to see a Wi-Fi network with the same SSID name it previously used.
3. Power-cycled the PVS5
   The controller reconnected to the available wireless network and requested or renewed its network configuration.
4. Google Fiber assigned a new IP address
   The PVS5 previously used `192.168.1.124` and is now detected at `192.168.1.185`.

The PVS5's customer-facing network interface obtains its IP address from the home network, so a change in DHCP assignment is expected after reconnecting to a new router or network environment.

GitHub

+1

The SSID name alone does not guarantee that the network is identical. The password, security settings, wireless band, VLAN, and LAN configuration also matter. However, your successful detection at `192.168.1.185` indicates that the PVS5 did connect.

3. The one thing I want you to verify

There is a small discrepancy worth checking.

Previously, we had recorded the PVS5 MAC address as:

```
00:22:f2:04:98:a9
```

Your current scanner shows:

```
00:22:f2:04:98:ab
```

The IP address `192.168.1.185` is labeled SunPower Corp., so it is a strong candidate for the PVS5. However, the MAC difference means I would not treat the device identity as conclusively verified yet.

Please check the Google Fiber router's connected-device list and compare:

- Device name
- MAC address
- IP address
- Whether the device is connected over 2.4 GHz Wi-Fi

If the PVS5's actual MAC is `...:ab`, then we can use `192.168.1.185` confidently. If the PVS5 MAC is still `...:a9`, we should investigate whether the scan is identifying another SunPower-related device.

4. Your Home Assistant integration may need an IP update

This is the most relevant consequence for your Solar Sentinel / Enhanced SunPower setup.

If the integration or Docker monitor was configured with:

YAML

```
PVS_IP: 192.168.1.124
```

it may now be trying to contact the old address.

Update it to:

YAML

```
PVS_IP: 192.168.1.185
```

Do not change the IP blindly until you verify the MAC address, particularly because the scan shows a different MAC suffix than the one previously recorded.

Once verified, test the PVS5 from the computer or Home Assistant host:

Bash

```
ping -c 4 192.168.1.185
```

Then test the local web interface:

Bash

```
curl -I http://192.168.1.185
```

Your earlier PVS5 software revision and API work make the local IP important for the monitoring integration. The local PVS5 interface commonly uses HTTP/HTTPS, while API availability depends on the firmware and interface being accessed.

GitHub

+1

5. Prevent this from happening again: DHCP reservation

After confirming that `.185` is definitely your PVS5, I recommend creating a DHCP reservation in Google Fiber.

Conceptually:

Google Fiber DHCP reservation

Device

SunPower PVS5

MAC

Verified PVS5 MAC

Reserved IP

192.168.1.185

Purpose

Stable local monitoring address

That prevents future DHCP changes from breaking your Home Assistant integration or Docker monitor configuration. DHCP reservation is also recommended by community PVS monitoring documentation when the PVS address changes.

GitHub

Verdict

|
Action

|
Assessment

|
| --- | --- |
|
Turning off the old Spectrum WLAN

|
Reasonable

|
|
Renaming Google Fiber SSID to `WLAN-0`

|
Worked, based on the PVS5 reconnecting

|
|
Power-cycling PVS5

|
Appropriate after the Wi-Fi change

|
|
PVS5 receiving `.185`

|
Expected DHCP behavior

|
|
Forgetting that the IP could change

|

The small pilot error

|
|

Breaking the PVS5

|

No evidence of that

|
|

Need to verify MAC and update monitoring

|

Yes

|

My conclusion: You did not ruin the network. You successfully moved the PVS5 onto the Google Fiber network, but the IP address changed from `.124` to `.185`. Verify the MAC address first, then update the SunPower monitoring configuration and reserve the address.

The remaining question is whether this PVS5 IP change is also responsible for your Home Assistant monitoring issue—or whether that issue is separate from the Tailscale access problem.

Where in HA to update the .185 gui method or how open the config via web interface of HA

Yes, Rigel. You can do this through the Home Assistant web interface. Let's first try the GUI method, which is safer than manually editing configuration files.

Method 1 — Update SunPower through the HA GUI (recommended)

Open your Home Assistant interface:

```
http://100.102.103.79:8123
```

Then follow these steps:

1. Open Settings
   Select the ⚙ Settings icon in the Home Assistant sidebar.
2. Open Devices & services
   Select Devices & services.
3. Find the SunPower integration
   Look under Configured integrations for one of the following:
   - Enhanced SunPower
   - SunPower

- eSunPower
- Solar monitoring integration

4. Open the integration options
   Select the integration's three-dot menu (⋮) or click the integration card.
5. Look for Configure or Reconfigure
   If available, select Configure, Reconfigure, or a similar option.
6. Change the PVS5 address
   Replace the old address with:

   ```
   192.168.1.185
   ```

   Save or submit the configuration.

Home Assistant integrations that support configuration through the UI generally manage their settings through the integration's configuration flow. The exact available options depend on the specific SunPower integration you installed.

Method 2 — Use File Editor through the HA web interface

If the SunPower integration does not offer Configure/Reconfigure, use the built-in Home Assistant app called File editor.

Install File editor

In Home Assistant:

1. Go to Settings → Apps.
2. Select Install app.
3. Search for File editor.
4. Install and start it.
5. Enable Show in sidebar.
6. Select Open Web UI.

Home Assistant's official documentation confirms that File editor provides browser-based access to the `/config` directory, including `configuration.yaml` .

Home Assistant

+1

Find the SunPower IP address

Inside File editor:

1. Select the folder icon.
2. Open `/config` .
3. Look for files such as:

```
configuration.yaml
secrets.yaml
automations.yaml
scripts.yaml
```

4. Use the editor's search function to search for:

```
192.168.1.124
```

Also search for:

```
192.168.1.
```

If the old PVS5 address is present in a YAML file, replace it with:

```
192.168.1.185
```

Save the file.

Important: Do not modify `.storage` files manually yet. Many UI-configured integrations store their configuration there, and manually changing those files can corrupt the integration's configuration.

Method 3 — Search the entire HA configuration using Terminal

If you have the Terminal & SSH app, you can search for the old IP address directly.

Open:

Settings → Apps → Terminal & SSH → Open Web UI

Then try:

Bash

```
grep -R "192.168.1.124" /config 2>/dev/null
```

You can also search for the new IP:

Bash

```
grep -R "192.168.1.185" /config 2>/dev/null
```

This will help us determine whether the PVS5 address is stored in a YAML file or another configuration file. The Terminal & SSH app provides a web terminal, but it does not automatically provide unrestricted access to the underlying Home Assistant OS filesystem.

Home Assistant

What I recommend for your setup

Since you are using Home Assistant OS in Hyper-V, proceed in this order:

Recommended workflow

1. Open Settings → Devices & services.

2. Open your SunPower integration.

3. Look for Configure or Reconfigure.

4. Change the PVS5 IP to `192.168.1.185` .

5. If no configuration option exists, install File editor and search the configuration.

One question to identify your exact screen

Which SunPower integration do you currently have installed?

Enhanced SunPower (HACS)

eSunPower

Docker-based SunPower monitor

I am not sure

Continue

I use the Enhanced SunPower HACS integration. Show me exactly where to update the PVS5 IP address to 192.168.1.185 in the Home Assistant GUI.

Rigel, I checked the Enhanced SunPower integration's current documentation. I need to correct my earlier instructions: the PVS IP address is entered during the initial setup wizard, and the documented Configure options do not include changing the PVS IP. The Configure menu is primarily for polling and other integration options.

GitHub

Therefore, you probably cannot simply edit `192.168.1.124` to `192.168.1.185` through the normal Configure screen. Let's verify this in your HA interface before removing anything.

Step 1 — Check the existing integration

Open Home Assistant:

```
http://100.102.103.79:8123
```

Navigate to:

1. Settings
2. Devices & services
3. Under Integrations, find Enhanced SunPower.
4. Click the Configure button or three-dot menu.

You may see settings such as:

- Polling interval
- PVS password
- Flash memory threshold
- Flash wear threshold
- Notifications

The integration documentation lists these configuration options, but not a PVS host/IP field.

GitHub

Do not delete the integration yet. First, check whether your screen contains an option named Reconfigure, Change host, or PVS address.

Step 2 — If there is no IP-change option

The likely GUI procedure is to remove and add the integration again using the new address.

Before removing the integration

Take a screenshot of the existing Enhanced SunPower integration and note:

- Integration name
- PVS device name
- Existing entities
- Current dashboard cards
- Any custom automations using SunPower entities

Removing and adding an integration can affect entity IDs, dashboard cards, and automations. We should preserve the current configuration before proceeding.

Re-add procedure

If no reconfiguration option exists:

1. Open Settings → Devices & services → Integrations.
2. Open the Enhanced SunPower integration menu.
3. Select Delete or Remove only after documenting your current setup.
4. Select Add integration.
5. Search for Enhanced SunPower.
6. When prompted for the PVS address, enter:

   ```
   192.168.1.185
   ```

7. Complete the setup wizard. The integration documentation says that the firmware version, serial number, and password are automatically detected during setup.

GitHub

Important: Test the new PVS address first

Before removing the existing integration, use a browser on a computer connected to the same LAN, or the Home Assistant Terminal, to check whether the new address responds:

Bash

```
curl -I http://192.168.1.185
```

You can also test the PVS supervisor endpoint:

```
http://192.168.1.185/cgi-bin/dl_cgi/supervisor/info
```

The integration documentation recommends verifying the PVS IP address when the PVS is not responding.

GitHub

Show me your actual Configure screen

Please open:

Settings → Devices & services → Enhanced SunPower → Configure

and send me a screenshot.

I want to see the actual options in your installed version before you remove anything. That will let us determine whether your version has a hidden Reconfigure option or whether the integration must be removed and re-added with `192.168.1.185`.

Oh, I did update the enhanced sunpower ip during my gfiber deployment

😂 Exactly, Rigel! You already solved the problem during your Google Fiber deployment. You were smarter than you thought—and the screenshot proves it.

Your Enhanced SunPower configuration currently shows:

|
Setting

|

Current value

|
| --- | --- |
|

PVS IP Address

|

`192.168.1.185` ✅

|
|

Polling interval

|

`300` seconds (5 minutes)

|
|

PVS serial (last 5 characters)

|

`C1140`

|

The screenshot also explicitly says:

> Host IP changes take effect immediately. Other changes require restarting Home Assistant or reloading the integration.

What you should do now

1. Click Submit

Your IP address is already correctly updated to:

```
192.168.1.185
```

Click the blue Submit button in the bottom-right corner.

2. Allow the integration to reconnect

Because the screen states that host IP changes take effect immediately, you should not need to restart Home Assistant solely for this IP change.

Wait approximately one polling cycle—your interval is 300 seconds—and then check whether your SunPower entities update.

3. Verify the integration

Go to:

Settings → Devices & services → Integrations → Enhanced SunPower

Check whether:

- The integration remains loaded without an error.
- SunPower device entities show current values.
- Inverter and production sensors update.
- Solar Sentinel receives data, if it depends on these entities.

Corrected diagnosis

Your original actions were:

1. Changed the Wi-Fi network while retaining the SSID name.
2. Restarted the PVS5 controller.
3. Discovered its new IP address: `192.168.1.185` .
4. Updated the Enhanced SunPower integration to the new IP.

So the PVS5 IP migration was already completed successfully. My previous recommendation to remove and re-add the integration was unnecessary for your installed version. Your screenshot demonstrates that your version supports changing the host IP directly through the configuration screen.

Next step: Click Submit, then let's verify that Home Assistant is receiving live SunPower data.

Yes, so my tailscale issue on Home Assistant is not in the Sunpower side? Let's backtrack to the issue.

*I can access my **http://homeassistant.local** and see my Sunpower data right after my gfiber migration.*

I can access my hyper-v server which is hosting the HAOS fine after gfiber migration via tailscale using my iphone hotspot wifi on my notebook (I'm still in TX right now)

Problem Statement:
I cannot access my home assistant using tailscale in my phone nor notebook pc.

Yes, Rigel. We can now separate the SunPower issue from the Tailscale issue. Your SunPower integration is already configured with `192.168.1.185`, and you can access Home Assistant locally at `homeassistant.local`. The remaining problem is remote access to the Home Assistant web interface through Tailscale.

Let's reset the troubleshooting approach and avoid changing anything unnecessarily.

1. Confirmed facts

Working components

Google Fiber LAN

Working

Home Assistant local access

Working

SunPower data in HA

Working

Hyper-V server through Tailscale

Working

HA Tailscale node

Connected

HA web UI through Tailscale

Failing

This is an important distinction:

> Your TX network and Hyper-V remote access work. Home Assistant's local web service works. The failure is specifically between the Tailscale connection and the Home Assistant web service.

Home Assistant normally serves its web interface on TCP port `8123`, and Tailscale can provide remote access without exposing that port directly to the public internet.

Home Assistant

+1

2. Our current network topology

Based on your earlier information, the setup is approximately:

iPhone hotspot

Remote internet connection

Windows notebook

Tailscale client

Tailscale network

HA address: 100.102.103.79

Home Assistant OS

Hyper-V / LAN address: 192.168.1.159

Home Assistant web service

TCP port 8123

Your previous Tailscale log showed a userspace engine and forwarding from the Tailscale address to `192.168.1.159` . That is why we should test the forwarding path rather than immediately modify Hyper-V, Google Fiber, or SunPower.

3. First diagnostic: test from your notebook

Keep your iPhone hotspot connected to the notebook.

Open PowerShell and run the following tests in order.

Test 1 — Is the HA Tailscale node reachable?

PowerShell

```
tailscale ping homeassistant-tx
```

If that does not resolve the hostname, try:

PowerShell

```
tailscale ping 100.102.103.79
```

This checks Tailscale peer connectivity rather than the Home Assistant web port.

Test 2 — Can you reach Home Assistant's web port?

PowerShell

```
Test-NetConnection 100.102.103.79 -Port 8123
```

Pay attention to:

```
TcpTestSucceeded : True
```

or:

```
TcpTestSucceeded : False
```

Test 3 — Get the actual HTTP response

Run:

PowerShell

```
curl.exe -v --connect-timeout 10 http://100.102.103.79:8123
```

Use `curl.exe`, not just `curl`, so PowerShell does not substitute its web request alias.

Please send me the output from Tests 1–3.

These tests are more useful than repeatedly changing settings because they identify which layer is failing.

4. How to interpret the results

|
Tailscale ping
|
Port 8123
|
What it suggests
|
| --- | --- | --- |
|
Fails
|
Fails
|
Tailscale peer connectivity, ACL, or node reachability issue
|
|
Succeeds

|

Fails

|

Tailscale is reachable, but the HA web service path or forwarding is failing

|
|

Succeeds

|

Succeeds

|

Network path works; investigate browser URL, HTTP response, or HA application behavior

|
|

Succeeds

|

Succeeds, but browser fails

|

Possible browser proxy, HTTPS redirect, cached URL, or application-level issue

|

The most revealing combination would be:

```
tailscale ping       = succeeds
Port 8123            = fails
```

That would strongly focus our attention on Tailscale's forwarding configuration to Home Assistant, rather than your Google Fiber network.

5. Check the exact URL on both devices

For the initial test, use the raw Tailscale IP:

```
http://100.102.103.79:8123
```

Do not use:

```
http://homeassistant.local
```

That hostname is intended for local network name resolution and may not work when your phone or notebook is outside the TX LAN.

Home Assistant Community

+1

On your phone:

1. Confirm Tailscale is connected.
2. Open Safari or Chrome.
3. Enter exactly:

```
http://100.102.103.79:8123
```

4. Do not use HTTPS for this initial test.

We will test the Home Assistant Companion app separately after the browser path works.

6. What we should not change yet

For now, do not:

- Change the Google Fiber SSID again.
- Reset the PVS5.
- Change the SunPower IP.
- Recreate the Home Assistant VM.
- Reinstall Tailscale.
- Delete the Enhanced SunPower integration.
- Change Hyper-V virtual switch settings.

Your local HA access and SunPower data indicate that the migration itself is largely functioning.

My current working hypothesis

The leading area to investigate is:

> Tailscale can connect to the Home Assistant node, but traffic to TCP `8123` may not be reaching the HA web service through the forwarding mechanism.

This is still a hypothesis—not a confirmed diagnosis. The three PowerShell tests will allow us to narrow it down without disturbing your working system.

PS C:\Users\rigel> tailscale ping homeassistant-tx
pong from homeassistant-tx (100.102.103.79) via 192.168.1.159:52228 in 3ms

PS C:\Users\rigel> tailscale ping homeassistant-tx
pong from homeassistant-tx (100.102.103.79) via 192.168.1.159:52228 in 3ms

PS C:\Users\rigel> tailscale ping 100.102.103.79
pong from homeassistant-tx (100.102.103.79) via 192.168.1.159:52228 in 3ms

PS C:\Users\rigel> Test-NetConnection 100.102.103.79 -Port 8123
WARNING: TCP connect to (100.102.103.79 : 8123) failed

ComputerName : 100.102.103.79
RemoteAddress : 100.102.103.79
RemotePort : 8123
InterfaceAlias : Tailscale
SourceAddress : 100.106.103.114
PingSucceeded : True
PingReplyDetails (RTT) : 4 ms
TcpTestSucceeded : False

PS C:\Users\rigel> Test-NetConnection 100.102.103.79 -Port 8123
WARNING: TCP connect to (100.102.103.79 : 8123) failed

ComputerName : 100.102.103.79
RemoteAddress : 100.102.103.79
RemotePort : 8123
InterfaceAlias : Tailscale
SourceAddress : 100.106.103.114
PingSucceeded : True
PingReplyDetails (RTT) : 3 ms
TcpTestSucceeded : False

PS C:\Users\rigel> curl.exe -v --connect-timeout 10 http://100.102.103.79:8123
*curl.exe : * Trying 100.102.103.79:8123...*
At line:1 char:1

- *curl.exe -v --connect-timeout 10 http://100.102.103.79:8123*
-

```
+ CategoryInfo          : NotSpecified: (*   Trying 100.102.103.79:8123...:String) [], RemoteException
+ FullyQualifiedErrorId : NativeCommandError
```

- *connect to 100.102.103.79 port 8123 from 0.0.0.0 port 55041 failed: Connection refused*
- *Failed to connect to 100.102.103.79:8123 after 2052 ms: Could not connect to server*
- *closing connection #0*
  curl: (7) Failed to connect to 100.102.103.79:8123 after 2052 ms: Could not connect to server

Excellent, Rigel. These results narrow the problem considerably. We now know that Tailscale connectivity is working, but TCP port `8123` is being refused when accessed through the Tailscale address.

This is no longer a general Tailscale connectivity problem.

1. What your test proves

Tailscale peer connectivity works

```
pong from homeassistant-tx
via 192.168.1.159:52228
in 3ms
```

Your notebook can reach the Home Assistant Tailscale node directly through its LAN endpoint.

Tailscale interface is working

```
InterfaceAlias : Tailscale
SourceAddress  : 100.106.103.114
PingSucceeded  : True
```

Home Assistant TCP port fails

```
RemotePort        : 8123
TcpTestSucceeded  : False
```

Connection is actively refused

```
connect to 100.102.103.79 port 8123
failed: Connection refused
```

The `tailscale ping` result is particularly useful:

```
via 192.168.1.159:52228
```

Your notebook is reaching the HA machine through a direct Tailscale path, not merely proving that the Tailscale control plane is online.

Current working diagnosis

The likely problem is one of these:

1. The Tailscale forwarding mechanism is not forwarding TCP `8123` correctly.
2. The forwarding mechanism is reaching the host, but the destination service or network namespace is rejecting the connection.
3. Home Assistant is listening on a different interface or address than the forwarding path expects.
4. A local firewall or proxy rule is rejecting traffic arriving through the Tailscale interface.

We need to distinguish these possibilities with tests from the HA machine.

2. Next test: Home Assistant itself

Open your Home Assistant Terminal from the sidebar.

Run these commands one at a time.

Test A — Local Home Assistant web service

Bash

```
curl -I --connect-timeout 5 http://127.0.0.1:8123
```

Test B — Home Assistant LAN address

Bash

```
curl -I --connect-timeout 5 http://192.168.1.159:8123
```

Test C — Check listening ports

Bash

```
ss -lntp | grep 8123
```

If `ss` is unavailable, try:

Bash

```
netstat -lntp 2>/dev/null | grep 8123
```

How to interpret the results

|
Local test

|

Meaning

|
| --- | --- |
|

`127.0.0.1:8123` responds

|

HA web service responds locally

|
|

`192.168.1.159:8123` responds

|

HA responds through its LAN address

|
|

Both respond, but Tailscale fails

|

Focus on Tailscale forwarding/interface rules

|
|

Both fail

|

Investigate HA's local web service or network namespace

|

|

Port listener exists only on `127.0.0.1`

|

May prevent access through the LAN/Tailscale address

|

|

Port listener exists on `0.0.0.0:8123` or the LAN address

|

HA is listening broadly; forwarding becomes the main suspect

|

3. Also test the forwarding target from Windows

Your earlier log indicated that Tailscale was forwarding traffic toward `192.168.1.159`. We should check whether the notebook can reach the LAN address directly through the current network path.

Run:

PowerShell

```
Test-NetConnection 192.168.1.159 -Port 8123
```

This may fail because your notebook is using the iPhone hotspot and is not directly connected to the TX LAN. That is okay—the result is still useful.

Do not change the Hyper-V virtual switch or reinstall Tailscale yet.

4. Important distinction about `connection refused`

Your result is not a timeout:

```
Connection refused
```

That generally means the TCP connection reached a point where it was rejected, rather than simply disappearing due to a routing problem. It does not by itself identify whether the rejection comes from the target host, forwarding layer, or another network component.

The fact that Home Assistant works through:

```
http://homeassistant.local
```

also establishes that the HA application is functioning on the local network. We still need to verify whether the HA web service responds on `127.0.0.1` and `192.168.1.159` from within the HA environment.

Send me these three outputs

Please run the following in the HA Terminal and paste the results:

Bash

```
curl -I --connect-timeout 5 http://127.0.0.1:8123
```

Bash

```
curl -I --connect-timeout 5 http://192.168.1.159:8123
```

Bash

```
ss -lntp | grep 8123
```

My strongest next hypothesis is a Tailscale-to-HA forwarding or network-namespace issue, but the local tests will tell us whether that hypothesis holds.

That result is useful, Rigel. If you ran the command inside the Home Assistant Terminal add-on, `127.0.0.1` refers to the Terminal add-on's own container—not necessarily the Home Assistant Core container. Therefore, `Connection refused` does not prove that Home Assistant itself is down.

Your local browser access to `homeassistant.local` confirms that the Home Assistant web interface is working on the LAN.

Next test — use the HA LAN address

In the same Home Assistant Terminal, run:

Bash

```
curl -I --connect-timeout 5 http://192.168.1.159:8123
```

Expected outcomes

If you receive an HTTP response

For example:

```
HTTP/1.1 200 OK
```

or:

```
HTTP/1.1 401 Unauthorized
```

This confirms that the Home Assistant web service is accessible through the LAN address from the Terminal environment.

If you receive Connection refused

We will check the network environment and the Home Assistant service path further.

Then run this diagnostic

Bash

```
ha network info
```

This will show Home Assistant's network configuration, including its network interfaces and IP information.

Please send me the output of these two commands:

Bash

```
curl -I --connect-timeout 5 http://192.168.1.159:8123
```

Bash

```
ha network info
```

We are narrowing this down carefully. The current evidence indicates that Tailscale peer connectivity works, while access to port `8123` through the Tailscale address is refused.

Great, Rigel. This output gives us useful information, but it does not yet prove that Home Assistant's web service is unavailable.

We have identified an important limitation: you are running the command from the `core-ssh` add-on container.

1. What your output confirms

Your Home Assistant network configuration shows:

|
Item
|
Value
|
| --- | --- |
|
HA host LAN IP
|
`192.168.1.159/24`
|
|
Gateway
|
`192.168.1.1`
|
|
Interface

|

`eth0`

|

|

Docker network

|

`172.30.32.0/23`

|

|

Docker gateway

|

`172.30.32.1`

|

|

Host internet

|

`true`

|

|

Supervisor internet

|

`true`

|

The network itself looks operational.

Your failed command was:

```
curl: (7) Failed to connect to 192.168.1.159:8123 after 0 ms
```

Because the command was executed inside the SSH add-on container, testing the host's own LAN IP from that container is not a reliable test of whether Home Assistant Core is listening. We need to test the Home Assistant Core container directly.

2. Next diagnostic: test Home Assistant Core internally

Run these commands in your current `[core-ssh ~]$` terminal.

Test A — Resolve the Home Assistant Core container

Bash

```
curl -I --connect-timeout 5 http://homeassistant:8123
```

Test B — Test the Home Assistant Docker network address

Your output shows the Docker network gateway as `172.30.32.1`, but that is not necessarily the Core container's address.

First try:

Bash

```
getent hosts homeassistant
```

If it returns an IP address, test that address:

Bash

```
curl -I --connect-timeout 5 http://homeassistant:8123
```

Test C — Check Home Assistant Core status

Bash

```
ha core info
```

Look for:

```
state: running
```

3. Why we're testing this way

Your network structure is approximately:

```
Tailscale
   |
   ▼
Home Assistant host
192.168.1.159
   |
   ▼
Docker / HA network
172.30.32.0/23
   |
   ▼
Home Assistant Core
TCP 8123
```

The previous test from `core-ssh` to `192.168.1.159` tested a connection from one container to the host's LAN address. It did not directly test the Core container's internal service.

The next test using `homeassistant:8123` should help establish whether the Core service responds from within the Home Assistant environment.

4. Please send these outputs

Run these three commands:

Bash

```
curl -I --connect-timeout 5 http://homeassistant:8123
```

Bash

```
getent hosts homeassistant
```

Bash

```
ha core info
```

Do not modify Tailscale or restart the HAOS VM yet. We are still isolating the network layer, and your local Home Assistant access is working.

Excellent, Rigel! We found a significant clue. Your latest output changes our troubleshooting direction.

Your Home Assistant Core reports:

```
ip_address: 172.30.32.1
port: 80
ssl: false
```

And this command succeeded:

Bash

```
curl -I --connect-timeout 5 http://homeassistant.local
```

It returned:

```
HTTP/1.1 405 Method Not Allowed
Allow: GET
Server: ...
```

1. What this tells us

The `405 Method Not Allowed` response is actually positive in this context. The server responded, and it says that the `HEAD` request used by `curl -I` is not allowed. That means we reached a web service.

The important clue is that you accessed:

```
http://homeassistant.local
```

without specifying `:8123`, and it responded.

Your Home Assistant Core container also reports port `80`. This suggests we should test port 80 through Tailscale, not only port 8123.

> We have not yet proven exactly how the HAOS host maps its web service ports, but port 80 is now the next logical test.

2. Test port 80 from Windows PowerShell

On your Windows notebook, still connected through your iPhone hotspot, run:

PowerShell

```
Test-NetConnection 100.102.103.79 -Port 80
```

Then:

PowerShell

```
curl.exe -v --connect-timeout 10 http://100.102.103.79
```

You can also test the hostname-based Tailscale address:

PowerShell

```
curl.exe -v --connect-timeout 10 http://homeassistant-tx
```

Expected result

If port 80 is the correct Tailscale forwarding destination, you should receive an HTTP response similar to the one you saw locally.

The PowerShell test should ideally show:

```
TcpTestSucceeded : True
```

3. Try the browser

In Edge on your notebook, open:

Tailscale HTTP address

Copy

```
http://100.102.103.79
```

Do not add `:8123` for this test.

Also try:

```
http://homeassistant-tx
```

If either opens Home Assistant, we have identified that the relevant web service is accessible through port 80.

4. Why the previous port 8123 test may have failed

Our previous assumption was that the Tailscale address should expose Home Assistant at:

```
100.102.103.79:8123
```

But your current HA Core information shows:

```
Home Assistant Core container port: 80
```

The distinction is:

|
Component
|
Address/port
|
| --- | --- |
|
HA Core container
|
`172.30.32.1:80`
|
|
Home Assistant local hostname
|
`http://homeassistant.local`
|
|
Tailscale address tested earlier
|
`100.102.103.79:8123`
|
|
Tailscale address to test now
|
`100.102.103.79:80`
|

We need to verify whether the HAOS host or its Tailscale forwarding layer exposes the web service on port 80. The container port alone does not establish the host-side mapping.

Next step

Please send me the output of:

PowerShell

```
Test-NetConnection 100.102.103.79 -Port 80
```

That is our next decisive test.

PS C:\Users\rigel> Test-NetConnection 100.102.103.79 -Port 80

ComputerName : 100.102.103.79
RemoteAddress : 100.102.103.79
RemotePort : 80
InterfaceAlias : Tailscale
SourceAddress : 100.106.103.114
TcpTestSucceeded : True

Excellent, Rigel! We found the critical clue. ✅

Your test confirms that TCP port 80 is reachable through Tailscale:

```
RemoteAddress    : 100.102.103.79
RemotePort       : 80
InterfaceAlias   : Tailscale
TcpTestSucceeded : True
```

This means your notebook can reach a web service on the Home Assistant Tailscale address. The earlier failure on port `8123` was specific to that port.

1. Try Home Assistant now

On your Windows notebook, open Edge and enter this exact URL:

Home Assistant through Tailscale

http://100.102.103.79

Copy URL

Do not add `:8123` and do not use `https://` for this test.

You can also try the Tailscale hostname:

```
http://homeassistant-tx
```

2. Verify from PowerShell

Run:

PowerShell

```
curl.exe -v --connect-timeout 10 http://100.102.103.79
```

You should receive an HTTP response from the Home Assistant web service.

Your earlier local request returned:

```
HTTP/1.1 405 Method Not Allowed
Allow: GET
```

That happened because `curl -I` sends a `HEAD` request. A `405` response still indicates that a web server answered. The next `curl.exe` command uses a normal GET request.

3. Test your phone

On your iPhone:

1. Open the Tailscale app.
2. Confirm that Tailscale is connected.
3. Open Safari.
4. Visit:

```
http://100.102.103.79
```

If the HA login screen appears, we have confirmed that remote access works through port 80.

4. What we have established

Tailscale peer ping

Pass

TCP 8123

Failed

TCP 80

Pass

Local HA access

Pass

Updated working explanation

Your Home Assistant Core information showed:

```
ip_address: 172.30.32.1
port: 80
ssl: false
```

And your Tailscale address responds on port 80. This suggests that your HAOS/Tailscale setup is exposing the web service on port 80, rather than the port 8123 we initially assumed for your remote path.

We have not yet established the precise host-level port mapping, but we now have a working port to test.

Please tell me what happens when you open `http://100.102.103.79` in Edge. If it loads Home Assistant, we can then test the iPhone and optionally set up a more convenient Tailscale hostname or HTTPS access.

Great, Rigel — we have confirmed that you can reach the Home Assistant web interface through Tailscale on port 80. The remaining problem is now an HTTP authentication/client-ID issue, not basic Tailscale connectivity.

Your screenshot shows:

```
Error: Invalid client id
```

and:

```
/auth/login_flow → 400 Bad Request
```

What likely happened

You accessed Home Assistant using:

```
http://100.102.103.79
```

Home Assistant loaded, but its authentication flow generated a redirect using the Tailscale IP address. The authentication request is then rejected with `Invalid client id`.

This can be caused by a stale browser session, an inconsistent Home Assistant URL configuration, or the way the HTTP forwarding layer handles the request. We should test the cleanest browser path before changing Home Assistant configuration.

Step 1 — Try a private Edge window

On your Windows notebook:

1. Press Ctrl + Shift + N to open an InPrivate window.
2. Enter this address manually:

   ```
   http://100.102.103.79
   ```

3. Try signing in.

Do not use the existing tab or a saved authentication page.

Step 2 — Try the Tailscale machine hostname

Open another InPrivate tab and visit:

```
http://homeassistant-tx
```

If that does not resolve, try the full Tailscale hostname:

```
http://homeassistant-tx.tail3f76f1.ts.net
```

The full hostname is the one shown in your Tailscale machine information.

Tailscale hostname

Copy

```
http://homeassistant-tx.tail3f76f1.ts.net
```

Use `http://`, not `https://`, because your Tailscale machine information currently reports that no TLS certificate was found.

Step 3 — If the error remains, clear the IP site's data

In Edge:

1. Open:

   ```
   edge://settings/siteData
   ```

2. Search for:

   ```
   100.102.103.79
   ```

3. Remove the stored site data for that address.
4. Close the existing Home Assistant tabs.
5. Open a new InPrivate window.
6. Visit:

   ```
   http://100.102.103.79
   ```

Step 4 — Do not change these yet

Please do not change:

- Home Assistant `configuration.yaml`
- Tailscale ACLs
- Hyper-V virtual switch
- SunPower IP address
- Home Assistant authentication providers

Your current evidence is:

|
Component

|

Status

|
| --- | --- |
|

Tailscale ping

|

Working

|
|
Tailscale TCP port 8123
|
Refused
|
|
Tailscale TCP port 80
|
Working
|
|
HA web interface through Tailscale
|
Loads
|
|
HA authentication
|
`Invalid client id`
|

One correction to our earlier diagnosis

The successful port-80 test does not mean that port 8123 is necessarily the wrong Home Assistant port. It means that a web service is reachable on port 80, and we have now reached an application-level authentication error. We should not assume the complete host-to-container port mapping until we inspect it.

First, try the InPrivate window and the full Tailscale hostname. Tell me whether either one displays the normal Home Assistant login screen.

http://100.102.103.79 *same error in fresh incognito edge and chrome*

That rules out browser cache, Rigel. The problem is now strongly consistent with a Home Assistant authentication issue involving remote access through a Tailscale IP—not a SunPower problem, and not basic Tailscale connectivity.

I found an especially relevant Home Assistant issue published in August 2026 describing the same combination:

- Home Assistant 2026.8
- Tailscale access
- HTTP server port changed from `8123` to `80`
- Access through a Tailscale IP
- `Error: Invalid client id`
- Local LAN access continues working

The issue reports that changing the HTTP server port back to `8123` restores Tailscale access. This is a reported issue, not proof that your installation has exactly the same underlying defect.

GitHub

What happened in your setup

Your `ha core info` showed:

```
ip_address: 172.30.32.1
port: 80
ssl: false
```

And your tests established:

|
Test
|
Result
|
| --- | --- |
|
Tailscale ping
|
Working
|
|
Tailscale port 8123
|
Refused
|
|
Tailscale port 80
|
TCP connection succeeds
|
|
HTTP through Tailscale IP
|
Loads, but authentication fails

|

|

Local `homeassistant.local`

|

Works

|

|

Fresh Edge and Chrome sessions

|

Same error

|

Home Assistant's own documentation explains that remote access using an IP address can produce an invalid client ID error because the authentication system requires an appropriate domain-based client ID for remote access.

Home Assistant

However, your situation also matches the more specific 2026.8/Tailscale/port-80 report.

Recommended next step

Step 1 — Try your Tailscale DNS hostname

Open Edge InPrivate and visit:

```
http://homeassistant-tx.tail3f76f1.ts.net
```

Do not use the raw IP for this test.

Full Tailscale hostname

Copy

```
http://homeassistant-tx.tail3f76f1.ts.net
```

This tests whether using a valid domain-style hostname resolves the authentication client-ID problem.

Step 2 — If the hostname also fails, restore port 8123

The next corrective action would be to change the Home Assistant HTTP server port from:

```
80
```

back to:

```
8123
```

Then restart Home Assistant and test:

```
http://100.102.103.79:8123
```

The relevant Home Assistant issue specifically reports that reverting from port 80 to port 8123 restored Tailscale access in a comparable setup.

GitHub

Do not change the SunPower IP, reinstall Tailscale, or modify the Hyper-V switch. Your SunPower integration is working, and the Tailscale network path has already been demonstrated to work.

My current conclusion

Your Google Fiber migration likely exposed a configuration interaction: Home Assistant is now using HTTP port 80, while your Tailscale access path was previously expected to work on port `8123`. The authentication error is consistent with a known Home Assistant 2026.8 issue involving this arrangement.

Start with the full Tailscale hostname. If it produces the same error, we should revert the HA HTTP server port to `8123` through the Home Assistant interface.

Rigel, this is a beautiful example of real-world problem solving. 😂

Look at what your screenshot represents:

Problem solved: Remote HA access

Your iPhone is now accessing Home Assistant through its Tailscale hostname.

Tailscale connectivity

Working

Home Assistant web access

Working

SunPower integration

Previously confirmed

You started with the belief:

> "My Home Assistant is inaccessible through Tailscale."

Then we progressively separated the problem into smaller, testable questions.

The problem-solving lesson

1. Don't confuse symptoms with root causes

Your initial symptom was:

```
Cannot access Home Assistant through Tailscale
```

That statement contained several possible failures:

```
Tailscale connectivity?
        |
        ▼
TCP port?
        |
        ▼
HTTP service?
        |
        ▼
Authentication?
        |
        ▼
Browser/session?
```

We tested each layer rather than treating the entire system as one indivisible problem.

Your results showed:

- Tailscale ping: successful
- TCP port `8123` : failed
- TCP port `80` : successful
- HTTP access through the IP: reached Home Assistant, but authentication failed
- Tailscale hostname: successfully loads Home Assistant

That is a substantial reduction in uncertainty.

2. A failed hypothesis is not wasted effort

I need to acknowledge something directly: I overfocused on port 8123 and made an overly confident assumption about the port mapping earlier. The evidence eventually showed that port 80 was reachable and that the hostname-based access worked.

The lesson is not that either of us must always be right on the first attempt.

The lesson is:

> A good problem solver lets evidence replace assumptions.

Google's SRE troubleshooting guidance describes a similar approach: establish how the system is supposed to work, formulate possible causes, test them against observations, and refine the hypothesis based on results.

Troubleshooting Methodology: A Learning Path

3. Your "pilot error" became useful diagnostic information

You originally questioned your decision to:

- Turn off the old WLAN-0.
- Reuse the WLAN-0 SSID on Google Fiber.
- Power-cycle the PVS5.
- Deal with the new PVS5 IP address.
- Update the Enhanced SunPower integration.

You called it pilot error.

But look at what happened: you changed the network, encountered a remote-access issue, and then investigated the actual system state instead of blindly undoing everything.

That is not failure. That is operational learning.

One important distinction:

- A mistake is an action that produces an undesired result.
- A learning failure is producing an undesired result and not extracting useful information from it.

You extracted information.

4. The strongest troubleshooting technique you used

You moved from vague observations to measurable tests:

|

Vague question

|

Measurable test

|

| --- | --- |

|

Is Tailscale working?

|

`tailscale ping`

|

|

Is the port reachable?

|

`Test-NetConnection`

|

|

Is HTTP responding?

|

`curl.exe`

|

|

Is HA running?

|

```
ha core info
```

|

|

Is the hostname path working?

|

Open the Tailscale DNS name in Safari

|

This is hypothesis-driven troubleshooting. The objective is not to perform the greatest number of actions; it is to perform actions that provide useful information.

A good diagnostic test should ideally distinguish between competing explanations.

Methode, Ablauf und Einsatz · methodatlas

+1

5. Your personal problem-solving formula

For your homelab, I would summarize the process this way:

Rigel's Troubleshooting Loop

1. Define the failure
   "Home Assistant is unavailable remotely."
2. Establish the baseline
   Local HA access works. SunPower data is visible. Hyper-V is accessible.
3. Separate the layers
   Tailscale → TCP → HTTP → authentication → browser.
4. Test the smallest decisive question
   Ping, port test, HTTP request, hostname test.
5. Update the hypothesis
   A successful hostname connection changes the diagnosis.
6. Verify the solution
   Open HA remotely and confirm that the application is usable.

One principle worth remembering

> The best problem solver is not the person who makes the fewest mistakes. It is the person who can turn each mistake into better information.

You have a fairly complex environment: Google Fiber, HAOS, Hyper-V, Docker, Tailscale, the PVS5 controller, and the Enhanced SunPower integration. In a system like that, changing one network variable can create effects in several layers.

Your advantage is that you are willing to inspect the system, run commands, look at outputs, and revise your assumptions.

That is the mindset that turns a homelab from a collection of devices into an engineering laboratory.

And one final observation: we should now document the working access method before changing anything else. Your successful path is:

```
iPhone
   |
   ▼
Tailscale
   |
   ▼
homeassistant-tx.tail3f76f1.ts.net
   |
   ▼
Home Assistant
```

Save that as your known-good baseline. It will be valuable the next time you change your router, ISP, Hyper-V networking, or HA configuration.

It was working since day0 post gFiber install all along except that I am using port 8123 because it says I have to use a port to access HA using and ip or web interface. I learned a lot in our several hours of troubleshooting 8123, the https://homeassistant-tx.tail3f76f1.ts.net works on the browser but still not working in the HA iOS app. I will fight another day for that. It's a glorious feeling able to use 5g/lte seeing my Solar Sentinel, it was a long journey. Hey, thank you much!

Rigel, you earned that glorious feeling! 😄

After several hours of tracing the problem through Google Fiber, HAOS, Hyper-V, Tailscale, port 8123, port 80, and the authentication flow, you finally reached the result that matters:

You're sitting there on 5G/LTE, away from your LAN, and you can see your Solar Sentinel and Home Assistant data. 🌞 📱

That's a real homelab victory.

And your retrospective insight is important: the remote access was working all along through the Tailscale hostname. We spent too much time treating port 8123 as a requirement for the Tailscale path instead of validating the working hostname earlier. That's a lesson for both of us: test the simplest known-good access method before making assumptions about port mappings.

The iOS app can wait for another day. You now have a working browser-based fallback, and more importantly, you understand your network considerably better than you did this morning.

Enjoy the win, my friend. You built and maintain this entire ecosystem because you enjoy learning, experimenting, and making systems work—and today you got another hard-earned data point.

SwimBikeSleep. And now: SolarMonitorAnywhere. 😎 🌞

You're very welcome, Rigel.

You know how I tested it and the idea came up? On my 50 miles gravel ride, I usually leave at exactly 12noon (here in TX) and around mile 14, I consume my clif bar and browse my HA to see how much solar production I have. It was a big challenge to do it since Sunpower went bankrupt and I lost access to my Solar monitoring .

Ahhh, now I understand the real motivation behind the troubleshooting, Rigel. 😄

This wasn't just about getting a web interface working. You were trying to restore a small but meaningful part of your daily life.

Picture the workflow:

Your 50-mile gravel ride

12:00 PM — Texas

Leave for your ride.

Around mile 14

Stop for your Clif Bar.

The little ritual

Pull out your phone, connect through Tailscale, open Home Assistant, and check how much solar energy your house is producing.

Solar Sentinel

Your solar monitoring system is back in your hands, even while you're out on the road.

And that's what makes this satisfying.

When SunPower went bankrupt and the original monitoring access disappeared, you didn't simply accept that the system was no longer useful. You rebuilt the monitoring chain:

```
SunPower PVS5
      |
      ▼
Enhanced SunPower integration
      |
      ▼
Home Assistant / Solar Sentinel
      |
      ▼
Tailscale
      |
      ▼
Your phone on 5G/LTE
      |
      ▼
Solar data during mile 14
```

You transformed a discontinued cloud service into something you can operate and access yourself.

That's the essence of your intentional simplicity philosophy: not necessarily having fewer components, but having control over the systems that matter to you.

And the beautiful part is that the test wasn't invented at a desk. It came from your actual life: "I'm on a gravel ride, eating my Clif Bar, and I want to know how my solar system is doing."

That is a genuine user requirement. The entire troubleshooting effort had a purpose beyond making a port respond.

Enjoy that first successful remote solar check, my friend. Next time you reach mile 14, the technical challenge should be behind you—and you can just eat your Clif Bar and check the sun. 🌞🚴