07 - RouteTrack Pi — Automated Route Processing (systemd Service + Timer)

Date: December 25, 2025
Category: Raspberry Pi / GPS / Automation / systemd
Backlink: 06 – RouteTrack Pi — Route Processing & Summary Generation

---

Project Goal

This phase automates RouteTrack's daily processing workflow so I don't have to manually run the processor.

Because the GPS logger continuously writes to SQLite, the route processor must run with **exclusive database access**. The automation is designed to safely:

1. Stop `routetrack-logger.service` (release DB lock)
2. Run `routetrack-process.py` using the venv Python
3. Restart `routetrack-logger.service`
4. Log all output to systemd journal for review

This creates a repeatable, production-style "ingest → process → report" pipeline.

Why systemd Timer (instead of cron)

I used a systemd timer because it provides:

- Reliable scheduling with clear “next run” info
- Centralized logs via `journalctl`
- Easy enable/disable and status checks
- Clean separation of concerns (service runs once; timer schedules it)

Create Processor Wrapper Script

This wrapper script is the “safe runner” that prevents SQLite lock errors.

Create the file

```
sudo nano /opt/routetrack/bin/routetrack-run-processor.sh
```

Script used

```
#!/usr/bin/env bash
set -euo pipefail

# ------------------------------------------------------------
# RouteTrack - Safe Processor Runner
# ------------------------------------------------------------
# Stops the GPS logger (releases SQLite locks),
# runs the daily processor, then starts the logger again.
# ------------------------------------------------------------

LOGGER_SERVICE="routetrack-logger.service"
PROCESSOR="/opt/routetrack/bin/routetrack-process.py"
PYTHON="/opt/routetrack/venv/bin/python"
```

```
echo "$(date -Is) RouteTrack processor wrapper starting..."

echo "$(date -Is) Stopping ${LOGGER_SERVICE}..."
systemctl stop "${LOGGER_SERVICE}"

# small pause so file handles release cleanly
sleep 2

echo "$(date -Is) Running route processor..."
"${PYTHON}" "${PROCESSOR}"

echo "$(date -Is) Starting ${LOGGER_SERVICE}..."
systemctl start "${LOGGER_SERVICE}"

echo "$(date -Is) RouteTrack processor wrapper completed."
```

Make executable:

```
sudo chmod +x /opt/routetrack/bin/routetrack-run-processor.sh
```

Create the systemd Service (oneshot)

The systemd service runs the wrapper script one time.

Create the unit file

```
sudo nano /etc/systemd/system/routetrack-processor.service
```

Unit file used

```
[Unit]
Description=RouteTrack Daily Route Processor
Wants=network-online.target
```

```
After=network-online.target

[Service]
Type=oneshot
User=root
Group=root
ExecStart=/opt/routetrack/bin/routetrack-run-processor.sh
StandardOutput=journal
StandardError=journal
```

Reload units:

```
sudo systemctl daemon-reload
```

Test the Service Manually

Run it once to confirm it works:

```
sudo systemctl start routetrack-processor.service
```

View the logs:

```
journalctl -u routetrack-processor.service --no-pager -l
```

Confirm Logger Restarted Properly

```
systemctl status routetrack-logger.service --no-pager -l
```

The logger returned to an **active (running)** state immediately after processing.

Create the systemd Timer

The timer schedules the processor to run automatically every day.

Create the timer file

```
sudo nano /etc/systemd/system/routetrack-processor.timer
```

Timer used (daily at 2:10 AM local time)

```
[Unit]
Description=Run RouteTrack Processor Daily

[Timer]
OnCalendar=*-*-* 02:10:00
Persistent=true
RandomizedDelaySec=30
Unit=routetrack-processor.service

[Install]
WantedBy=timers.target
```

Reload and enable:

```
sudo systemctl daemon-reload
sudo systemctl enable --now routetrack-processor.timer
```

Verify the Timer is Active

Show timers:

```
systemctl list-timers --all | grep routetrack
```

Check timer status:

```
systemctl status routetrack-processor.timer --no-pager -l
```

This confirms:

- Timer is **active (waiting)**
- Next trigger time is scheduled

- It triggers `routetrack-processor.service`

Confirm Data Was Written

After a successful run, verify the summary table:

```
sqlite3 /opt/routetrack/data/routetrack.sqlite \
"SELECT * FROM daily_summary ORDER BY date DESC LIMIT 3;"
```

This confirms the processor populated:

- `daily_summary`
- `stop_events`

and is producing real computed metrics.

Run On Demand (Manual Trigger Block)

This is the clean “run it right now” workflow (and verify it worked) without touching the timer schedule.

Run the processor service now

```
sudo systemctl start routetrack-processor.service
```

View the last 50 log lines from the run

```
journalctl -u routetrack-processor.service -n 50 --no-pager -l
```

Confirm the next scheduled timer run is still set

```
systemctl list-timers --all | grep routetrack
```

Operational Notes

Why the logger must stop

SQLite needs exclusive access for derived-table regeneration (`DELETE` + `INSERT`).
Running the processor while the logger is inserting can produce:

- `sqlite3.OperationalError: database is locked`

This wrapper workflow prevents that completely.

Where logs live

Everything is stored in systemd journal:

```
journalctl -u routetrack-processor.service --since "today" --no-pager -l
```

Next Steps

Now that processing is automated daily, the next phase is the dashboard:

- Flask web service (local)
- Leaflet map page
- Route drawing from `gps_points`
- Stop markers from `stop_events`
- Daily totals from `daily_summary`

Revision #2
Created 25 December 2025 15:45:09 by Nate Nash
Updated 25 December 2025 21:16:19 by Nate Nash