DROSJER QUANTUM

Orbital Intelligence Report #001

OMM Migration Readiness Audit for SatNOGS Open Source Ground Station Network

Target: /private/tmp/satnogs-db **Generated:** 2026-04-30T19:06:48.244793+00:00
Engagement ID: c88060019a31

> *This sample report was generated from a real audit of the SatNOGS open-source satellite tracking project, AGPL 3.0 licensed. Findings, dependencies, and remediation recommendations shown here are actual output from the Drosjer Quantum audit toolkit, provided for demonstration purposes. Customer engagements follow the same methodology applied to the customer's private codebase under confidentiality.*

What This Audit Covers

- **Dependency Inventory** — Every TLE / catalog-width / Alpha-5 / format-lock-in touchpoint, with file:line and severity.
- **Element Set Quality Assessment** — Per-object freshness, update-frequency, consistency, and drag-stability grades.
- **Impact Assessment** — Severity-tiered findings (critical / high / medium / low) with remediation specificity.
- **Remediation Plan** — Sequenced code changes, library upgrades, and schema modifications.
- **Validation Checklist** — Post-migration verification steps the engineering team runs before production cutover.

Each tier has defined source-code volume and revision-pass limits. Small: ≤50K lines, 1 revision. Medium: ≤250K lines, 2 revisions. Complex: custom scope, 2+ revisions.

Why This Matters

Deadline:	approximately July 20, 2026 (81 days)
Current max catalog ID:	68837
Ceiling:	69,999 (not 99,999)
IDs remaining:	1162

Executive Summary

This audit evaluates **SatNOGS Open Source Ground Station Network** for dependency on the TLE format that the US Space Surveillance Network is scheduled to exhaust on approximately **July 20, 2026** — **81 days** from the report date (2026-04-30). The scanner examined **169 source files** and surfaced **17 migration-relevant findings**, distributed as 0 critical, 5 high, 11 medium, and 1 low-severity. Applied against the weighted-penalty model (detailed below), the codebase earns an OMM Migration Readiness grade of **B** with a score of **70/100**.

B

70 / 100

Score = 100 – (20 × critical) – (7 × high) – (2 × medium) – (0.5 × low), floored at 0, with per-severity caps (critical –70, high –20, medium –10, low –5). Your score breaks down as: **100 – (5×7 capped at 20) – (11×2 capped at 10) – (1×0.5) = 70**.

Minor remediation required before July 20, 2026.

Minor remediation required before July 20, 2026. Recommended sequencing: resolve all critical findings first, then high-severity, then medium and low as engineering capacity permits. The *Recommended Remediation Sequencing* section below lists every critical and high-severity item with its exact file-and-line location plus a specific remediation. The *Findings Breakdown* and per-severity tables that follow the sequencing list enumerate the complete inventory.

Findings Breakdown

Severity	Count
CRITICAL	0
HIGH	5
MEDIUM	11
LOW	1

HIGH (5)

File:Line	Category	Pattern	Snippet	Remediation
db/api/tests.py:289	tle_parsing	TLE_FILE_LOAD	`"tlefile": "starlink.tle",`	Hardcoded .tle/.3le path; source OMM/JSON from CelesTrak or Space-Track.
db/api/tests.py:306	tle_parsing	TLE_FILE_LOAD	`"tlefile": "starlink.tle",`	Hardcoded .tle/.3le path; source OMM/JSON from CelesTrak or Space-Track.
db/base/tasks.py:96	tle_parsing	TLE_PARSE_TWOLINE2RV	`sgp4_tle = twoline2rv(tle[1], tle[2], wgs72)`	Migrate to OMM/JSON via sgp4.omm or Satrec.sgp4init with OMM fields.
db/base/tasks.py:108	tle_parsing	TLE_PARSE_TWOLINE2RV	`sgp4_last_tle = twoline2rv(`	Migrate to OMM/JSON via sgp4.omm or Satrec.sgp4init with OMM fields.

db/base/migrations/0022_add_tle_model.py:39	tle_parsing	TLE_FILE_LOAD	`bases=('base.tle',),`	Hardcoded .tle/.3le path; source OMM/JSON from CelesTrak or Space-Track.

MEDIUM (11)

File:Line	Category	Pattern	Snippet	Remediation
db/base/tasks.py:23	format_lock_in	FMT_TLE_ONLY_IMPORT	`from sgp4.io import twoline2rv`	Importing only the TLE parser locks downstream code to the TLE format.
db/base/models.py:561	catalog_width	CAT_LIMIT_99999	`validators=[MinValueValidator(-99999), MaxValueValidator(99999)],`	99999/100000 literal limit — Alpha-5 and 6-digit IDs now exist in the catalog.
db/base/models.py:561	catalog_width	CAT_LIMIT_99999	`validators=[MinValueValidator(-99999), MaxValueValidator(99999)],`	99999/100000 literal limit — Alpha-5 and 6-digit IDs now exist in the catalog.
db/base/models.py:587	catalog_width	CAT_LIMIT_99999	`validators=[MinValueValidator(-99999), MaxValueValidator(99999)],`	99999/100000 literal limit — Alpha-5 and 6-digit IDs now exist in the catalog.
db/base/models.py:587	catalog_width	CAT_LIMIT_99999	`validators=[MinValueValidator(-99999), MaxValueValidator(99999)],`	99999/100000 literal limit — Alpha-5 and 6-digit IDs now exist in the catalog.
db/base/migrations/0020_auto_20200802_1804.py:54	catalog_width	CAT_LIMIT_99999	`field=models.IntegerField(blank=True, help_text='Transmitter drift`	99999/100000 literal limit — Alpha-5 and 6-digit IDs now exist in the catalog.

			`from the published downlink frequency, stored in parts per billion (PPB)', null=True, validators=[django.core.validators.`	
db/base/migrations/0020_auto_20200802_1804.py:54	catalog_width	CAT_LIMIT_99999	`field=models.IntegerField(blank=True, help_text='Transmitter drift from the published downlink frequency, stored in parts per billion (PPB)', null=True, validators=[django.core.validators.`	99999/100000 literal limit — Alpha-5 and 6-digit IDs now exist in the catalog.
db/base/migrations/0020_auto_20200802_1804.py:89	catalog_width	CAT_LIMIT_99999	`field=models.IntegerField(blank=True, help_text='Receiver drift from the published uplink frequency, stored in parts per billion (PPB)', null=True, validators=[django.core.vali`	99999/100000 literal limit — Alpha-5 and 6-digit IDs now exist in the catalog.

			`dators.MinVa`	
db/base/migrations/0020_auto_20200802_1804.py:89	catalog_width	CAT_LIMIT_99999	`field=models.IntegerField(blank=True, help_text='Receiver drift from the published uplink frequency, stored in parts per billion (PPB)', null=True, validators=[django.core.validators.MinVa`	99999/100000 literal limit — Alpha-5 and 6-digit IDs now exist in the catalog.
db/base/fixtures/satelliteentries.json:7007	catalog_width	CAT_LIMIT_99999	`"norad_cat_id": 99999,`	99999/100000 literal limit — Alpha-5 and 6-digit IDs now exist in the catalog.
db/base/fixtures/transmitters.json:4135	catalog_width	CAT_LIMIT_99999	`"baud": 100000.0,`	99999/100000 literal limit — Alpha-5 and 6-digit IDs now exist in the catalog.

LOW **(1)**

File:Line	Category	Pattern	Snippet	Remediation
db/api/views.py:782	alpha5	ALPHA5_FIRST_CHAR_CHECK	`if any(x.isalpha() for x in lat):`	Alpha-5 first-char branch — handle 6-digit numeric IDs as well.

Pipeline Telemetry

What the Drosjer Quantum pipeline currently observes, captured at the time of this audit.

Metric	Value

Tracked satellites	30444
By orbital regime	DSP: 56 · GEO: 1495 · HEO: 1272 · LEO: 26356 · MEO: 1265
Element-set rows	405581
Conjunction records	1637 (1246 with PC)
Last CelesTrak sync	2026-04-30T18:01:14.797123+00:00
Last Space-Track sync	2026-04-25T04:30:02.782857+00:00
Graded satellites	59539
Grade distribution	A: 73 · B: 2286 · C: 18270 · D: 28559 · F: 10351

Recommended Remediation Sequencing

Tackle in this order. Critical findings gate every other item — fix first, then high-severity, then medium / low as capacity permits.

1. HIGH `db/api/tests.py:289` — Hardcoded .tle/.3le path; source OMM/JSON from CelesTrak or Space-Track.
2. HIGH `db/api/tests.py:306` — Hardcoded .tle/.3le path; source OMM/JSON from CelesTrak or Space-Track.
3. HIGH `db/base/tasks.py:96` — Migrate to OMM/JSON via sgp4.omm or Satrec.sgp4init with OMM fields.
4. HIGH `db/base/tasks.py:108` — Migrate to OMM/JSON via sgp4.omm or Satrec.sgp4init with OMM fields.
5. HIGH `db/base/migrations/0022_add_tle_model.py:39` — Hardcoded .tle/.3le path; source OMM/JSON from CelesTrak or Space-Track.

Appendix

Scanner summary

Files scanned	169
Scan started	2026-04-30T19:06:46.023079+00:00
Scan completed	2026-04-30T19:06:46.163389+00:00
Findings rendered	17
Internal / fixture findings excluded	0

Scanner methodology

The scanner applies regex patterns across four categories — TLE parsing, catalog width, Alpha-5 compatibility, and format lock-in — to source files with a whitelisted set of extensions. Context-window checks suppress common false positives (for example, a 5-character string slice is flagged only if `norad`, `cat`, or `satno` appears within three lines). The full finding catalogue with exact line numbers and remediation notes is in the sibling JSON file.