1) Airspace / restrictions / NOTAM / UTM feeds (EANS / lennuinfo)

Your own requirements mention integrating with Lennuinfosüsteem (NOTAM areas, flight restriction areas) and also NOTAM/airspace info as a key external input.

Eelanalüüs

NORTAL

Incoming SLA you actually need

- **Availability (monthly): e.g. 99.5–99.9% depending on how operationally "hard" you are.**
- **Freshness / max staleness: "published changes available within X minutes."**
- **Latency: typical and p95/p99 response time.**
- **Update semantics: do you get snapshots, deltas, or both? any "partial publish" risk?**
- **Incident comms: 24/7 contact + status notifications for outages or degraded data.**
- **Planned maintenance notice: minimum notice window + maintenance windows.**

Plus (not SLA, but required): interface contract

- **JSON schema / OpenAPI or GeoJSON conformance commitment + versioning + deprecation window (this is the "it won't deviate" part).**

2) Drone operator registry + permissions/authorisations (Transpordiamet)

Your functional list explicitly calls out integration with Transpordiameti droonikäitajate andmebaas.

Eelanalüüs

And the interview transcript reinforces the idea of checking if a flight is permitted / who is responsible etc.

Tannar 121225 Recording 355_res…

Incoming SLA you need (if used in live ops)

- **Availability: ideally high, but you can mitigate with caching (see below).**
- **Latency: especially if the UI does "click drone → show permit owner" in real time.**
- **Freshness: when a permit/registration is updated, how quickly does the API reflect it?**

- **Rate limits: so you don’t get surprise-throttled during incidents.**

Pragmatic twist: if they won’t give a strong SLA, ask for permission to cache + a bulk export (nightly snapshot) so outages don’t blind you.

3) “Other agencies feed you stuff” (environment / ministry datasets)

The transcript explicitly mentions consuming inputs from Transpordiamet, Keskkonnaamet, Kliimaministeerium.

Tannar 121225 Recording 355_res…

These are usually: protected areas, restrictions, permits, planned activities, etc. (The exact dataset depends on what you agree.)

Incoming SLA you need

- **Often not strict low-latency; it’s more about predictable publishing:**
    - **Publish schedule (daily/weekly) + max staleness**
    - **Bulk download reliability (if it’s file-based)**
    - **Change notifications (schema or semantics changes)**
- **Availability can be “best effort” *if* you can cache and tolerate being slightly stale.**

4) Geospatial services / map servers / restricted area layers (GIS)

Nortal explicitly lists “Georuumilised teenused (GIS, kaardiserverid, NOTAM-id, piiratud alad)” as a key integration family.

NORTAL

Incoming SLA you need

- **If basemap-only: you can cache aggressively → SLA can be softer.**
- **If “restricted areas” are operational logic: treat it like airspace restrictions:**
    - **Availability + freshness + change control**
- **Also ask for very specific geo guarantees:**
    - **coordinate system, geometry validity, max feature size, ID stability.**

5) Police / operations systems that can provide context (e.g., patrol location via KILP)

The transcript mentions KILP as a system used on patrol tablets and implies it can provide patrol location ("their system gives us location").

Tannar 121225 Recording 355_res…

Tannar 121225 Recording 355_res…

Incoming SLA you need (if you really depend on it)

- **Latency/jitter: location updates have to be "near real time" if used for decisioning.**
- **Availability during incidents (this one matters most exactly when everything else is on fire).**
- **Data quality: accuracy/precision bounds (even coarse), update rate, and what "unknown" looks like.**
- **Security + audit: who can see what, and proof you didn't leak it.**

(If it's "nice-to-have context", don't over-contract it — just build graceful degradation.)

6) Crisis-management / emergency platforms and law-enforcement operational systems

Nortal lists kriisijuhtimise platvormid and õiguskaitse operatiivsüsteemid as systems your platform may integrate with.

NORTAL

NORTAL

Incoming SLA you need

- **This is less about milliseconds and more about reliability + clarity:**
    - **Availability and support window**
    - **Message delivery guarantees (if event-based): at-least-once vs exactly-once, ordering, replay**
    - **Retention & audit requirements (often legally required)**

The "incoming SLA template" I'd reuse across all of the above

When you write the requirements, keep it boring and enforceable:

- **Availability % + how measured**
- **p95/p99 latency targets**

- **Max staleness / update propagation time**
- **Planned maintenance windows + notice**
- **Incident response times (acknowledge, workaround, fix) + comms channel**
- **Rate limits / fair use**
- **Security requirements (auth method, logging, audit)**
- **Change management: versioning + deprecation window + schema published**

That last line is the secret sauce: most breakages come from "format drift", which uptime SLAs don't cover at all.

---

1) Sensor vendors (inbound data) — *contract + interface spec + support SLA*

Your docs explicitly list planned sensors: radars, RF sensors, acoustic sensors, EO/IR cameras, RemoteID receivers, AI classification sensors.

Assurance you want:

- **Interface contract**: protocol + message schema (you already mention at least SAPIENT as a requirement), required metadata, timestamps, units, coordinate system, rate limits, event semantics.
- **Performance/quality SLOs**: latency, detection/event rate expectations, clock sync assumptions, duplicate/false alarm handling.
- **Support SLA**: firmware updates, bugfix windows, incident response, replacement/repair for hardware.

Why: sensor vendors love "minor firmware updates" that silently change fields, rates, or meanings.

2) Border surveillance / "piirivalvesüsteemid" (inbound) — *inter-agency data sharing + technical contract*

Both docs mention integrating with border-related systems ("piiriandmed", "piirivalvesüsteemid").

Assurance you want:

- **Data sharing agreement / MoU**: legal basis, classification, who can see what, retention, auditing, onward sharing.
- **Technical contract**: schema + versioning + change notice, plus availability expectations if it's operationally critical.

3) Radar networks operated by defense or aviation authorities (inbound) — *data contract + ops SLA*

Nortal's doc explicitly calls out "Radarivõrgud, mida haldavad kaitse- või lennundusasutused" and references **ASTERIX (Eurocontrol categories)**.

Assurance you want:

- **Standards commitment**: which ASTERIX categories/editions, any local extensions, update frequency.
- **Change process**: how upgrades are announced, test feeds, deprecation windows.
- **Ops SLA**: uptime, planned maintenance windows, incident comms (because outages here are not "meh").

4) Crisis management center platforms (inbound and outbound) — *data sharing + "interface SLA"*

You explicitly mention "kriisijuhtimiskeskuste platvormid" and "juhtimiskeskused".

Assurance you want:

- **Shared semantics**: what constitutes an "incident", "alert", "track", who owns the truth when data conflicts.
- **Interoperability contract**: schema, IDs, dedup rules, update / delete semantics, access control model.
- **Availability + support** if they rely on your picture during incidents (and vice versa).

5) Police / law enforcement operational systems (inbound/outbound) — *data sharing + strict audit requirements*

Nortal lists "Õiguskaitse operatiivsüsteemid" and your market-study mentions saving events "hilisemaks menetlemiseks".

Assurance you want:

- **Legal + audit contract**: logging requirements, evidentiary chain, retention, GDPR boundaries, who can query raw data vs. derived data.

- **Interface contract**: case/event IDs, export formats, time handling, redaction rules.

This is less “SLA” and more “if we don’t write this down, court will write it down for us.”

6) Geospatial services / map servers / restricted areas (inbound) — *format contract (WMS/WFS/tiles/GeoJSON) + stability rules*

Nortal explicitly calls out “Georuumilised teenused (GIS, kaardiserverid, NOTAM-id, piiratud alad)”.

Assurance you want:

- **CRS / coordinate guarantee** (WGS84 etc.), geometry validity rules, feature ID rules.
- **Published schemas** for properties, versioning, deprecation windows.
- **Caching/ETag/Last-Modified** (practical “assurance” so you don’t DDoS each other).

This is the same family as your NOTAM/UTM question—just broader.

7) Command & Control systems (C2) and defense interoperability (outbound) — *interface contract + classification/handling rules*

Nortal mentions a “C2 vaade”, “integratsioon C2 süsteemidega”, and NATO STANAG alignment.

Assurance you want:

- **Outbound data contract** (your “situational picture”): schema, security labeling, downgrade/redaction rules, transport profile.
- **Operational expectations**: what “near real-time” means, what happens under partial outage.

8) X-tee style integration / security-server patterns (transport layer) — *service + change policy*

Nortal literally says they prefer an “x-tee laadne lahendus”.

Assurance you want:

- Even if X-tee itself is standardized, you still need **producer/consumer agreements**: service definitions, error codes, auth requirements, versioning, and operational support expectations.

A useful rule of thumb: when is an SLA enough vs. when do you need a “data contract”?

- If you **only care it’s up** → SLA helps.

- If you care that **the payload stays parseable and meaningful** → you need an **interface/data contract** (schema + versioning + change management).
- If it involves **classified/personal data or operational authority** → you also need a **data-sharing/legal agreement** (roles, retention, audit, onward sharing).

For most of the items above (sensors, radar, GIS, crisis/police systems), the correct answer is: **SLA + data contract**, and sometimes **+ data-sharing agreement**.

A practical next step is to make a one-page inventory: **System / Owner / Inbound or Outbound / Criticality / What breaks if it changes / Required guarantees (SLA? schema? notice period?)**. That turns "we should probably…" into procurement-ready requirements without philosophical suffering.

For feeds like those two URLs, an ordinary SLA mostly answers the *wrong* question.

- **SLA** = "Is the endpoint up? how fast? what support window?"
- What you actually need is **an interface/data contract** = "What does the payload look like, what does it mean, and how/when can it change?"

Here's what I'd ask Lennuliiklusteenindus / EANS for, in practice.

1) A formal contract for the payload (not just "it's JSON")

For ...area24.json

Ask for one of:

- **OpenAPI spec** (if it's an API-ish JSON), or
- **JSON Schema** (best if it's "just a file feed"), including required/optional fields, types, enums, and date formats.

Also ask for:

- **Semantics** for key fields (units, time zones, coordinate reference, meaning of null/missing).

- **Backward compatibility rules** (e.g., “we only add fields; we don’t rename/remove without a version bump”).

For ...uas.geojson

GeoJSON has an actual standard: ask them to explicitly commit to **RFC 7946 compliance** (GeoJSON, WGS84 lon/lat, no weird CRS object).
Then ask for:

- Which **geometry types** can occur (Polygon/MultiPolygon/LineString/etc).
- Property schema for Feature.properties (again: JSON Schema helps even for GeoJSON properties).
- **ID stability** rules (do features have stable IDs across updates?).

2) Versioning + change management (this is the real assurance)

You want a written promise along these lines:

- **Versioned endpoints** (or at least versioned schema):
    - /v1/... stays compatible; breaking changes go to /v2/....
- **Deprecation policy**: breaking changes require (say) 3–6 months notice.
- **Change notifications**: a mailing list / webhook / status page where schema changes are announced.
- **Changelog**: human-readable release notes for payload changes.

If they refuse versioned URLs, insist on at least:

- A schemaVersion field in payload, and
- Guaranteed “no breaking change within a major version”.

3) Conformance tests / reference examples (more useful than an “official parser”)

An “official parser” can help, but it can also become a dependency trap. What you really want is:

- **Golden sample payloads** (normal case + edge cases)
- A **validation artifact**:
    - JSON Schema files published and versioned, *and/or*

    - A small **test suite** they run on their side before deployment ("provider contract tests")

Best-of-both-worlds ask:

- "Publish schema + 10–20 canonical examples + run schema validation in CI before releasing."

4) Data-quality guarantees (separate from format)

Format stability doesn't guarantee correctness. Consider asking for a lightweight "data SLO" like:

- Update frequency / max staleness (e.g., "updated within X minutes of AIS/UTM change")
- Error handling policy (what happens if upstream is missing?)
- "No partial writes" / atomic publish (either old snapshot or new snapshot, never half-written JSON)

5) Operational stuff that prevents nasty surprises

These make your integration calmer:

- Correct Content-Type (application/json, application/geo+json)
- ETag / Last-Modified so you can cache and avoid unnecessary pulls
- Rate limits and expected polling cadence
- Support contact + incident process
- If auth might appear later: commit to an auth plan (or at least advance notice)

What I'd put in a one-page "minimum acceptable" checklist

If you want a crisp ask, this is a good minimum:

1. **Published schema** (OpenAPI/JSON Schema) for each endpoint
2. **RFC 7946 compliance** for GeoJSON + documented allowed geometry types
3. **Backward compatibility rule**: additions only, no renames/removals without version bump
4. **Versioning + deprecation window** for breaking changes
5. **Change notification channel + changelog**
6. **Availability SLA** (uptime/latency/support), *separate section*

7. **Sample payloads + edge cases** (and ideally provider-side schema validation before deploy)

What you should still do on your side (because reality is a comedy)

Even with a contract, you'll sleep better if you:

- Validate against schema on ingest (fail fast + alert)
- Use a **tolerant parser** (ignore unknown fields; handle optional fields missing)
- Add a "quarantine" path: store raw payload when validation fails for debugging
- Monitor for drift (sudden field type changes, geometry explosions, etc.)

Bottom line: **SLA won't cover format guarantees unless you explicitly add an "interface SLA / data contract" section.** Ask for **schema + versioning + change process**; treat an "official parser" as optional frosting, not the foundation.

--

asutuse rollid kriisi ajal

pakkuda paar tsenaariumi

kuidas droonidega

--

olete mõelnud kuhu sensoreid paneksite?

mis oleks

mis nende poolt andmeturbe nõuded?

-

piir - läti piir?

kaamerad?

kuidas kaamerate võrk ehtud?

--

kellelt võiks saada segamise andmeid - kellel vaja?

--

kõik võimalikud liidestused millest hakkame sõltuma - nende SLA, backup

(map for example) - millise detailsusega kaarti üldse vaja.

kas saaksime oma kaardiserveri üldsegi?

-

kes hooldab erasensoreid, lepingud nendega, sla jne

-

--

lennuliiklus - teenuse SLA ja kirjeldus valideerimine; kas notamil validaator?

https://utm.eans.ee/avm/utm/uas.geojson

- mis teenuseid veel

--

Lennuliiklus – tuleviku uspace integratsioonid. Täna pole uav liiklust, aga kuidas SIIS järelvavet teostada. Automatiseerimine.

Mis siis kui remote-id näitab üht aga teeb teist (valetab)

Not phase 1, aga valmisolek cross-border u-space! Mida see peaks tähendama?

--

X tee ei sobi ilmselt oma latentsusega ka siis kui süsteemi kaudu püüdurdroonidele reaalajas sihtimisinfot peaksime edastama.

---

probability of detection vs. false-alarm – erinevatele kasutajatele erinev?

Eskalatsooniredel?

ED-322 | System Performance and Interoperability Requirements for Non-Cooperative UAS Detection Systems

????

https://www.eurocae.net/product/ed-322-system-performance-and-interoperability-requirements-for-non-cooperative-uas-detection-systems/

ED Interoperability Requirements for Counter-UAS systems – does not exist?

https://www.eurocae.net/working-group/wg-115/

kas lennuamet on liige?

BSI Flex 335 v2.0:2023, SAPIENT Network of autonomous sensors and effectors – Interface control document – Specification - Specification

---

How we avoid vendor lock-in

Non-open part , the issues with IP

WHY WE CANNOT USE C2 FROM ANY SYSTEM PROVIDER!!

Mis on viisid selle vältimiseks

Lisaks c2 ise arendamisele

Kogu integratsiooni koodi üleandmine … müük, avatud – muutmise loaga.

Kuidas kirjutada hanget /hinnata osalejaid

Kuidas nende piirangute juures hankida nii et ka oleks võimekaid tooteid.

Kuidas toime tulla “piiratud” seadmetega.

Litsensid

Reverse engineering protocols

Legal aspects.

--

--

Erinevate tootjate sensor fusion

Võimalusel prioriteet, mitte fusion?

Sõltuvalt olukorrast. Eskaleerimine

Kuidas teie asutuses näete eskalatsiooni.

Mis on need “kasutajad” keda peame jälgima (kommerts vs diy vs raadiovaikus)

--

mis käitumisplaan kui droon avastatakse?

(mida selle infoga peale hakatakse)

--

milline peaks olema käideldavus (internet maas)

-

MEREVÄGI - MEREPIIRI SPETSIIFIKA?

PLAANID DROONISEIREKS?

mida näete õiguslikult sellele süsteemile taksitusena

mis on teie suurim hirm selle süsteemi suhtes

(failure mode)

--

Vaadata erinevaid sensorite tootjate standardeid (spetsiifilis) ning analüüsida millised neist võiks millsite komplektidena sobida.

Et oleks modulaarne ja tuleviku laiendamiste kindel.

Mis võiks olla nõuded sensorite tarnijatele (uuendused, nii küber kui andmed)

Kuidas otsustada kas sensor fusion võtta tarnijalt või teha otsa ise

Samas:

How much future LLMs (implementation time 2+ years) reduce coding translation needs

Future proofing – we just need solid building blocks.

- Considering AI intelligence has been accelerating for couple of years – we cannot lock in. current sensor fusion is not what we will use. Todays so
- Hardware will be the limitation how much intelligence we can get out of it

-we can build proxys but need to be open standard.

--

Sensor fusion – how WHERE do we handle it? Ilmselgelt rohkem kui üks koht.

--

Mida võime skoobist välja lugeda?

Mida peaks teine faas täpselt saavutama?

Kas droonide andmebaas ühine või tarvis märkeid saada eraldi teha?

Tsoonide?

Mis infot tahaksite süsteemis kasutada, mis AK (v rohkem)

(mis on üldsegi AK ja mida see tähendab süsteemi jaoks)

Mis on erinevate osapoolte kasutusjuhud (oodatav koormus millega toime tulla) ja sellest tulenev optimaalne kasutajaliides. Palju on tundlikke andmeid mida ei soovi teiste kasutajatega jagada. Palju võib jagada tulemusi ilma andmeteta (drooni tuvastamine ilma sensori asukohata).

Mis on kellelgi salajane

Mis minimum mida hoida

Kas saate oma salajast ise hoida?

Kas vaja droone mida teised ei näe? – teeme musta ala

Et välja ei näita aga sensorid ikkagi talletavad andmebaasi?

Mis koordinaatsüsteemi peaks sisemiselt kasutama

Miks vaja buffer alasid?

Max kasutajate arv

Server sent events vs gRPC

Ramen ?

Special partitioning – uber

H3 hexagonal space index

--

--

Kuidas näitame ühte lendu üldse -kõik samal ajal, sensorid.

Milleks oleks mobiiliäpp hea

Kes näeb mis sensoreis v ala?

Kas piirame ala või sensorite järgi?