

Fleet Data Ownership & Security Handbook

A practical guide to data custody, access control, infrastructure responsibility, retention, backup, recovery, and security in self-hosted GPS tracking.

Self-hosting increases control. It also increases responsibility.

Published by Open VTS | Version 1.0

Publication information

Field	Detail
Publisher	Open VTS
Edition	2026 Edition
Version	1.0
Series	Open VTS Guide 06 of 06
Strategic job	Trust
Audience	CTOs, IT/security leaders, technical decision-makers, GPS/telematics business owners, system administrators, enterprise architects
Scope	Data control, shared security responsibility, recovery, monitoring, continuity, and verification in self-hosted GPS tracking

Editorial position

This handbook is a practical security and data-governance guide, not legal advice, a certification statement, or a claim that self-hosting is automatically secure. Requirements vary by jurisdiction, industry, customer obligations, and infrastructure design. Security-sensitive product behavior should be verified against the current Open VTS release and the organization's own production environment.

The central idea

Data ownership is not defined only by where a database is hosted. Real control depends on custody, access, retention, portability, security responsibilities, recovery capability, and the ability to continue operating when conditions change.

How to use this handbook

Use the eight-domain framework as a design and review checklist. For each domain, identify the owner, the control, the evidence, and the remaining risk. Where a control belongs to the customer, document the operational procedure rather than assuming the software vendor performs it. Where a control belongs to the product, verify the behavior in the current release rather than relying on a generic marketing statement.

Contents

PART I What Data Ownership Actually Means

Data estate - custody/control/access/rights - why server location is only one part of control

PART II The 8-Domain Fleet Data Control & Security Framework

Custody - Identity & Access - Infrastructure & Network - Application & Secrets - Data Lifecycle - Backup & Recovery - Monitoring & Accountability - Continuity & Exit

PART III The Shared Responsibility Model

Who secures what - operating cadence - incident and change discipline

PART IV Evaluate Open VTS

Open VTS data/security architecture - what the platform provides - what the operator must provide
- production questions

PART V Final Security & Data Control Checklist

A concise pre-production review for enterprise deployment

PART

What Data Ownership Actually Means

Begin by separating four concepts that are frequently compressed into one word: custody, control, access, and legal rights.

01 - Fleet Data Is More Than Location

A GPS platform does not manage only latitude and longitude. A serious fleet environment accumulates several classes of information that have different operational value, retention needs, sensitivity, and recovery importance. Treating all of them as one undifferentiated database makes governance harder.

THE FLEET DATA ESTATE

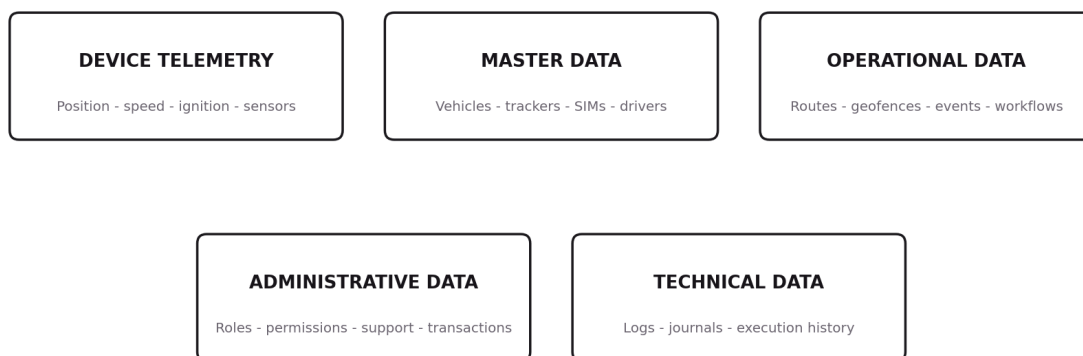


Figure 1 - The fleet data estate extends beyond raw telemetry.

Five useful data classes

Class	Typical examples	Why it matters
Device telemetry	Position, speed, heading, ignition, battery, sensors, alarms	High volume, time-sensitive, essential to history and operational truth.
Master data	Vehicles, trackers, SIMs, drivers, users	Defines identity and relationships across the platform.
Operational data	Geofences, routes, events, alerts, reports, workflows	Represents how the organization interprets and acts on telemetry.
Administrative data	Roles, permissions, plans, support, transactions, settings	Controls business administration and access boundaries.
Technical data	Logs, packet journals, execution histories, health records	Supports troubleshooting, accountability, reliability and recovery.

Protecting fleet data therefore means understanding the whole data estate. A policy that covers historical positions but ignores administrator access, workflow secrets, logs, or backup copies is incomplete.

02 - Custody, Control, Access, and Legal Rights

FOUR QUESTIONS THAT SHOULD NOT BE CONFUSED

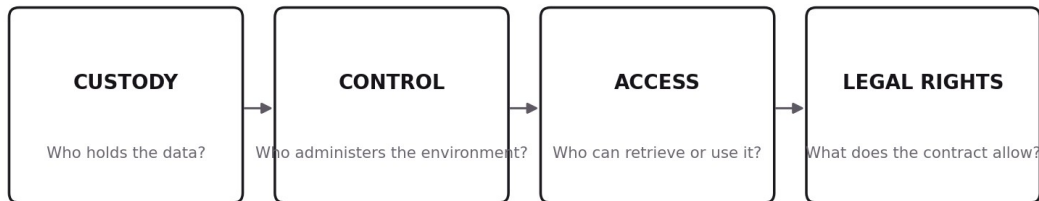


Figure 2 - These concepts overlap, but they are not interchangeable.

Custody

Custody asks who physically or logically holds the data. In a self-hosted model, important application and telemetry services may run in infrastructure controlled by the customer. A backup stored with a third-party provider, however, may have a different custody relationship than the production database.

Control

Control asks who administers the environment: who can change network rules, database configuration, retention, backups, server resources, or service credentials. Infrastructure control can be strategically valuable, but it creates operational responsibility.

Access

Access asks who can read or change information and through which mechanism: application role, API, direct database connection, server administration, backup system, or third-party integration. Data may be under customer custody while still being exposed through overly broad permissions.

Legal rights

Legal ownership and usage rights come from applicable agreements and law, not from server location alone. Self-hosting does not automatically transfer ownership of proprietary application source code. Open VTS core software is proprietary; the Open VTS mobile application is separately available with source code.

Customer-controlled infrastructure should not be confused with ownership of the application source code.

03 - Data Ownership Is a Lifecycle Question

A useful data-control review follows information from creation to end of life. Every stage introduces a different question about access, protection, retention, and responsibility.

FLEET DATA LIFECYCLE

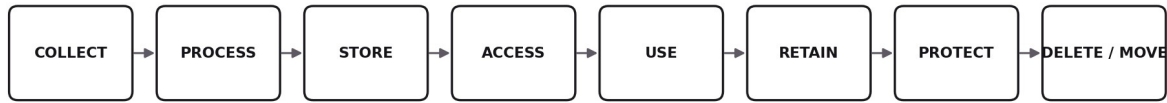


Figure 3 - The lifecycle provides a repeatable set of governance questions.

Stage	Primary question
Collect	What data is created by the device or user, and is it necessary?
Process	How is raw information decoded, normalized, or enriched?
Store	Where is the authoritative copy and which services depend on it?
Access	Who and what systems can retrieve or modify it?
Use	Which reports, workflows, APIs, or business processes consume it?
Retain	How long is it needed and what operational/regulatory purpose justifies that period?
Protect	Which security, backup, and monitoring controls apply?
Delete / Move	How is information removed, archived, or migrated when requirements change?

This lifecycle is deliberately independent of any specific product. It is the same discipline a buyer should apply to Open VTS, a SaaS platform, or a custom-built telematics system.

PART

The 8-Domain Fleet Data Control & Security Framework

The eight domains turn broad ideas such as “our data is secure” into specific responsibilities that can be assigned, tested, and evidenced.

8-DOMAIN FLEET DATA CONTROL & SECURITY FRAMEWORK

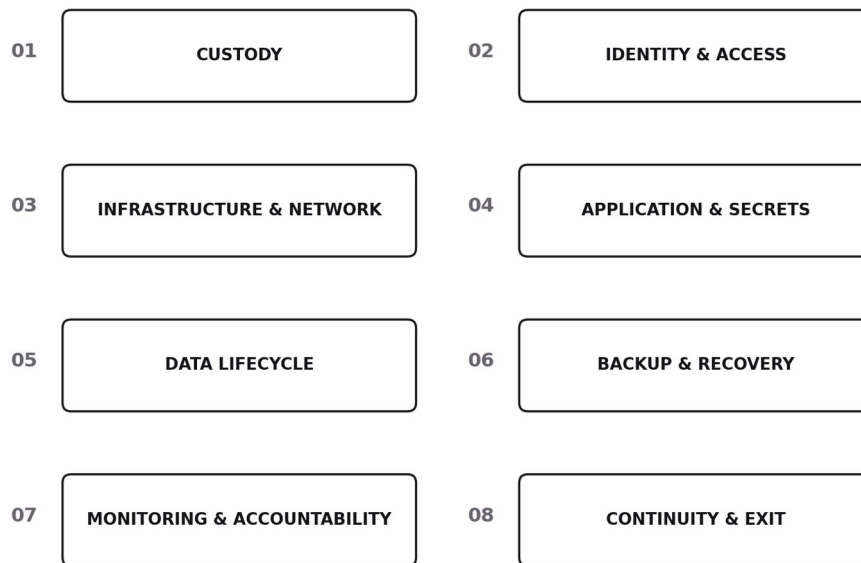


Figure 4 - Eight domains cover the control surface of a self-hosted tracking platform.

04 - Domain 1: Data Custody

Start by drawing the data boundary. Identify every service that stores, caches, forwards, or backs up important fleet information. This includes the primary database, telemetry/time-series storage, Redis or other live-state services, backup targets, log destinations, notification providers, map/geocoding services where used, and any integration that receives fleet information.

Questions to answer

- Where are application records stored?
- Where is historical telemetry stored?
- Where is current/live state maintained?
- Which services hold credentials or operational configuration?
- Which third parties receive fleet or user data?
- Who has infrastructure-level administrative access?
- Which data leaves the customer-controlled environment by design?

Open VTS perspective

Open VTS is architected around customer-deployed operational data services and provides structured access to fleet and application data through the platform and APIs. The current architecture includes PostgreSQL-based application/telemetry data services, Redis-backed operational roles, listener and backend services, and application/API clients. Buyers should still verify their exact backup, export, retention, integration, notification, and third-party service paths.

Data custody becomes useful only when the organization can name the systems, owners, and dependencies that hold the data.

05 - Domain 2: Identity & Access

Security begins with identity, but it does not end there. Authentication establishes who is making a request. Authorization determines what that identity is allowed to do. Scope determines where that permission applies. A mature platform needs all three concepts to remain clear as the organization grows.

AUTHORIZATION SHOULD NARROW ACCESS AT EACH STEP



Figure 5 - Effective authorization combines identity, responsibility, action, boundary, and resource.

Least privilege

Users should receive the minimum access required to perform their responsibilities. A fleet operator may need assigned vehicles and reports without financial configuration. A support team may need ticket/customer visibility without platform-wide administration. A customer user should not cross into another organization's resources simply because a direct API path exists.

Open VTS authorization model

Open VTS uses multiple role levels including Superadmin, Admin, Team, and User. Its Team authorization architecture also supports resource-scope concepts such as OWN, ASSIGNED, and TENANT for applicable capabilities, with backend enforcement rather than relying only on hidden navigation. Release-specific Team screens should still be verified during deployment and acceptance testing.

Access review questions

- Are permissions enforced on backend resources?
- Can vehicle access be limited to assigned or tenant scope?
- Can sensitive actions be separated from normal operational access?
- How are sessions/tokens revoked when a user leaves?
- Can denied access or privileged actions be investigated later?

06 - Privileged Administrative Access

Application permissions are only one layer. A person with server, cloud, database, DNS, backup, or superadmin access may have much broader authority than an ordinary application user. Production governance should therefore maintain a privileged-access register.

PRIVILEGED ACCESS REGISTER

SERVER / CLOUD	Who can administer the host?
DATABASE	Who can access operational data directly?
SUPERADMIN	Who can change platform-wide configuration?
DNS / TLS	Who controls public identity and certificates?
BACKUPS	Who can restore or delete recovery copies?

Figure 6 - Record who controls each privileged layer and why that access is required.

Privileged area	Minimum governance question	Recommended evidence
Server / cloud account	Who can start, stop, snapshot, or replace infrastructure?	Named accounts, MFA, access review.
Database	Who can query or alter production data directly?	Restricted accounts, audit/logging, documented purpose.
Superadmin	Who can change platform-wide configuration?	Named role holders, least privilege, change record.
DNS / TLS	Who controls domains and certificates?	Registrar/DNS ownership, renewal procedure.
Backups	Who can restore, export, or destroy recovery copies?	Backup console permissions, restore test, retention policy.

A production team should always know who has the power to bypass ordinary application permissions.

07 - Domain 3: Infrastructure & Network Security

A self-hosted tracking platform normally has two very different public traffic patterns: user/API traffic and device protocol traffic. They should be planned separately. Browser, mobile, and API traffic commonly enters through HTTPS. GPS devices may connect directly to dedicated TCP or UDP protocol ports. Databases, Redis, queues, and internal administration services should not be exposed publicly without a justified design requirement.

EXPOSE ONLY WHAT MUST BE REACHABLE

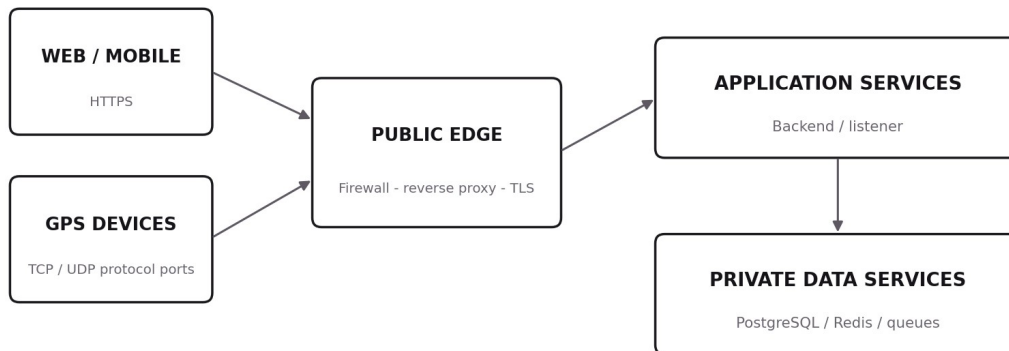


Figure 7 - Keep the public edge small and the internal service boundary explicit.

Core network principles

- Expose only ports required by users, APIs, and supported GPS protocols.
- Keep PostgreSQL, Redis, and internal service ports private unless a specific controlled access path is required.
- Restrict SSH/RDP or other server administration to trusted sources and named administrators.
- Use host/cloud firewall rules intentionally rather than relying on default exposure.
- Document protocol ports so device onboarding and incident investigation remain predictable.

A supported tracker cannot report if its TCP/UDP path is blocked; a secure design therefore balances necessary device reachability with minimal unnecessary exposure.

08 - TLS and Data in Transit

DATA IN TRANSIT DEPENDS ON THE PATH



Figure 8 - User/API traffic and GPS device traffic may have different transport-security capabilities.

Web, mobile, and API

HTTPS/TLS should protect normal browser and API traffic when the deployment is configured for public access. The operator is responsible for DNS, certificate deployment, private keys, renewal, and the network path that terminates TLS.

GPS device protocols

Tracker traffic depends on hardware and protocol capability. Many legacy GPS protocols were designed around plain TCP or UDP and may not support modern TLS. Security therefore cannot be described honestly with a blanket statement that "all telemetry is encrypted in transit." Buyers should

identify which device families support encrypted transport and compensate appropriately with network controls, endpoint validation, restricted listener exposure, device/server authentication features where available, and operational monitoring.

This is a hardware/protocol constraint as much as a platform constraint. Replacing or standardizing tracker hardware may sometimes be part of a long-term security improvement program.

09 - Domain 4: Application Security

Application security should be evaluated through implemented controls rather than adjectives. Useful questions include how passwords are stored, how sessions/tokens are handled, whether input is validated, whether permissions are enforced at backend boundaries, how sensitive integrations are protected, and how password-reset or other high-risk flows resist abuse.

Open VTS implementation evidence

Current Open VTS architecture includes controls around JWT-based access, bcrypt password hashing, backend role/permission guards, scoped Team authorization, request validation, listener-to-backend shared-secret authentication, reset-flow protections, encrypted third-party integration secrets, workflow-secret redaction/handling, and authorization-denial logging. Each control should be treated as one layer of the security model, not as proof that the surrounding server environment is secure by default.

Application review

Area	Verify
Authentication	Password storage, token/session lifecycle, reset protection, revocation.
Authorization	Backend enforcement, role boundaries, resource scope, sensitive actions.
Validation	Unexpected inputs rejected; request schemas and privilege checks enforced.
Secrets	Credentials not exposed in UI/logs; protected storage; rotation path.
Errors / logs	Useful diagnostics without leaking unnecessary secrets or sensitive data.

10 - Secrets & Credentials

A self-hosted platform can accumulate many secrets: database credentials, listener/backend shared secrets, API tokens, SMTP credentials, WhatsApp or messaging credentials, Firebase credentials, map/geocoding keys, cloud credentials, and integration tokens. These values should have a defined lifecycle rather than being copied informally between configuration files and chat messages.

SECRETS SHOULD HAVE A CONTROLLED LIFECYCLE



Figure 9 - Create, store, use, rotate, and revoke secrets deliberately.

Practical secret rules

- Issue only the credentials required for a specific service or integration.
- Restrict who can view or change secrets.
- Do not place secrets in screenshots, support tickets, source control, or general logs.
- Plan rotation before a credential is compromised.
- Revoke credentials when integrations, staff, or environments are retired.
- Use separate secrets for development/test and production wherever possible.

The question is not only whether a credential is encrypted at rest; it is whether the organization can identify where that secret exists and replace it safely.

11 - Domain 5: Data Lifecycle & Retention

Retention determines how much historical information remains available for reports, investigations, customer commitments, and business analysis. It also determines storage growth, backup size, recovery time, migration effort, and the amount of information exposed if a privileged account or backup is compromised.

RETENTION IS BOTH A SECURITY AND CAPACITY DECISION

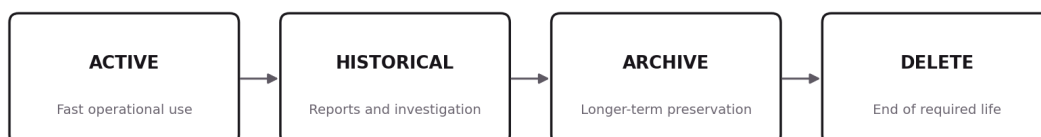


Figure 10 - Retention should move data deliberately from active use toward archive or deletion.

Not every dataset needs the same policy

Data class	Possible retention driver	Risk of keeping too long	Risk of deleting too early
Raw telemetry	Operational history, customer need, analysis	Large storage/backup footprint, larger exposure surface	Incomplete history, weak investigations/reports
Events / alerts	Operational review, safety, customer support	Unnecessary personal/operational history	Loss of audit/context
Packet journals / technical logs	Troubleshooting and recovery	High-volume technical data retained without purpose	Harder diagnosis of protocol incidents
Workflow / notification history	Automation audit and failure investigation	Credentials/context may require careful handling	Loss of execution evidence
Administrative/audit records	Governance and support	Accumulated sensitive activity history	Reduced accountability

Retention is a business decision, a security decision, and a capacity decision at the same time.

12 - Deletion & Data Minimization

Data minimization asks whether the organization needs to collect and retain every available field indefinitely. Removing information that no longer has a legitimate operational purpose can reduce storage, backup scope, migration effort, and security exposure. The correct policy depends on customer commitments, operational requirements, applicable law, and the ability to reproduce needed reports.

Deletion is not always instantaneous everywhere

Deleting a production record does not necessarily remove it immediately from backups, snapshots, archives, replicas, logs, or external systems that previously received it. A mature deletion policy therefore describes the lifecycle of copies, not only the primary database operation.

Questions to document

- Which roles may delete which data?
- Is deletion immediate, soft, delayed, or retention-driven?
- How long do backup copies remain?
- What happens to data already sent to an integration?
- Can identifiers be anonymized where operationally appropriate?
- What evidence confirms completion of the deletion policy?

13 - Domain 6: Backup & Recovery

Self-hosting gives the customer control over backup design, but it also makes backup quality an operational responsibility. A database backup alone may not be sufficient if the environment also depends on configuration, uploaded assets, TLS material, environment variables, integration credentials, or other deployment state.

BACKUP BECOMES RECOVERY ONLY AFTER RESTORE

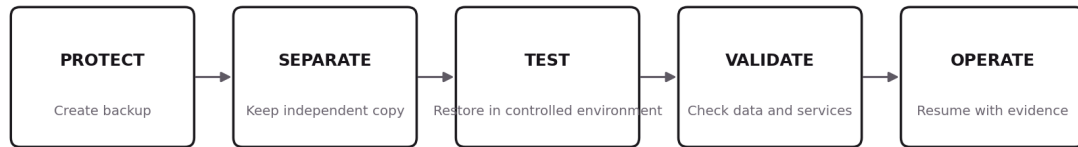


Figure 11 - Recovery requires a tested restore, not only a backup job.

RPO and RTO

Recovery Point Objective (RPO) expresses how much recent data the organization can afford to lose. Recovery Time Objective (RTO) expresses how long the platform can remain unavailable. These targets should drive backup frequency, off-server copies, restore procedures, and infrastructure design rather than being invented after an incident.

A backup is an assumption until it has been restored successfully.

Backup scope

- Application and telemetry databases
- Critical configuration and environment settings
- Uploaded files/assets required for operation
- TLS/domain material where appropriate and safely protected
- Integration configuration and secrets with appropriate security controls
- Documented deployment/version information so the restored environment can be reproduced

14 - Disaster Recovery

Disaster recovery is the process for restoring an acceptable operating state after a severe failure. Scenarios can include server loss, disk failure, database corruption, ransomware or credential compromise, accidental deletion, cloud instance loss, or a network event that isolates the service.

A practical recovery flow

1. Detect - establish what failed and whether compromise is suspected.
2. Contain - prevent additional damage, credential misuse, or corrupted writes.
3. Restore - rebuild infrastructure and recover known-good data/configuration.
4. Validate - confirm authentication, telemetry ingestion, reports, integrations, permissions, and monitoring.
5. Resume - re-open normal operation with heightened observation.
6. Review - document root cause, recovery performance, and corrective actions.

Recovery procedures should name decision-makers, backup locations, credentials required during emergency access, and the evidence used to declare the service healthy. The worst time to discover that only one person knows the restore procedure is during an outage.

15 - Domain 7: Monitoring & Accountability

Security controls lose value when failures remain invisible. Monitoring should make abnormal conditions observable across authentication, authorization, infrastructure, queues, databases, telemetry freshness, and integrations. Not every anomaly is a security incident, but unusual operational behavior often provides early evidence of one.

VISIBILITY CONNECTS SECURITY AND OPERATIONS

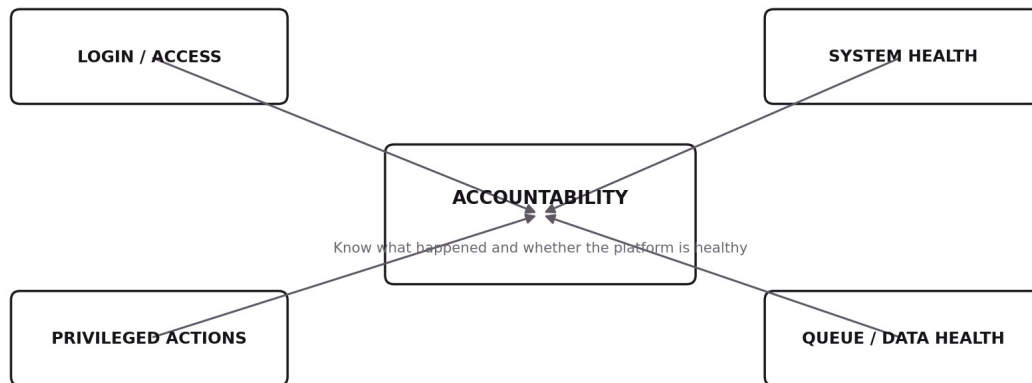


Figure 12 - Accountability requires visibility across both user activity and system health.

Signals worth watching

- Repeated login/reset failures or unusual privileged access
- Authorization denials and sensitive administrative actions
- CPU, memory, disk, network and process health
- Database/Redis availability and abnormal growth
- Queue backlog or failed workers
- Unexpected listener/protocol error patterns
- Large changes in online/offline/stale vehicle counts
- Integration/API failure spikes
- Backup failures and restore-test results

16 - Logs, Audit, and Evidence

The word “log” is often used for several different types of evidence. Separating them improves troubleshooting and governance.

Evidence type	Question it helps answer
Application logs	Why did a service or request fail?
Audit trail	Who changed or attempted to access a protected resource?
Device/event history	What did the vehicle/device report or what operational condition occurred?
Infrastructure logs/metrics	Was the server, network, database, queue, or runtime unhealthy?
Workflow/execution history	Which automation path ran, which action failed, and what was retried?

Retention and access to logs should be planned too. Logs can contain identifiers, URLs, error payloads, or other operational context. They should not become an uncontrolled secondary copy of sensitive data.

17 - Domain 8: Continuity & Exit

Security includes the ability to continue operating when conditions change. A platform may be perfectly healthy during normal operation but create serious risk if recovery, licensing, vendor dependency, or migration behavior is poorly understood.

Continuity questions

- What happens if the application server fails?
- What happens if a database or Redis service is unavailable?
- What happens to incoming telemetry during selected downstream failures?
- Which functions depend on external vendor or third-party services?
- What happens during temporary license-validation unavailability?
- How are devices redirected if the deployment IP/domain changes?
- What data and configuration are required to rebuild the platform elsewhere?

Exit questions

If the organization later replaces the platform, it should understand how to retrieve important records, preserve required history, reconfigure trackers, map users and identifiers, rebuild integrations, and satisfy contractual termination conditions. Customer-controlled infrastructure can simplify some aspects of continuity, but it should never be presented as automatic portability to every other software system.

Operational control should be tested during failure, not inferred from normal operation.

PART

The Shared Responsibility Model

A secure self-hosted deployment requires a clear boundary between what Open VTS provides in the software and what the customer must operate in the surrounding infrastructure.

18 - Who Secures What?

SELF-HOSTED SECURITY IS A SHARED RESPONSIBILITY

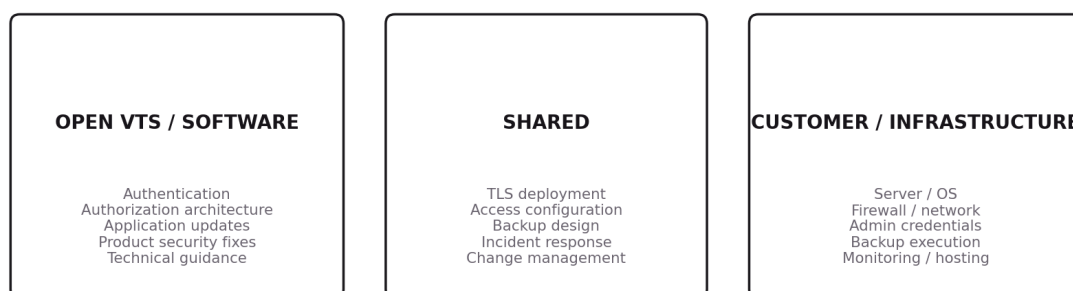


Figure 13 - Shared responsibility makes control boundaries explicit.

Open VTS / software side

- Application authentication and authorization mechanisms

- Backend permission enforcement and product-level validation
- Application updates and product vulnerability remediation
- Security-related product guidance and support
- Protection patterns for supported application secrets and integrations

Customer / infrastructure side

- Server and operating-system security
- Firewall and network exposure
- Cloud/hosting account security
- RDP/SSH and administrator credentials
- DNS and certificate operations
- Backup execution and restore testing
- Infrastructure monitoring and incident response

Shared responsibilities

Several controls require action from both sides. Open VTS may support HTTPS while the operator must configure DNS/TLS correctly. The product may provide roles while the operator must assign them correctly. The software vendor may release an update while the operator must schedule, deploy, and verify it. Backup architecture depends on product data boundaries while actual backup execution and restore testing belong to the deployment operation.

19 - Security Operating Cadence

Security becomes sustainable when recurring checks are embedded into normal operations rather than treated as one-time launch tasks.

Cadence	Suggested review
Daily	Platform/service health, backup result, critical queue failures, major authentication anomalies.
Weekly	Disk/database growth, stale/offline trends, failed integrations, listener/protocol error trends.
Monthly	OS/application patch review, privileged-access review, backup sample/restore check, secret expiration/renewal, retention health.
Quarterly	Full recovery exercise, access recertification, incident tabletop, credential rotation review, firewall/public exposure review, capacity/security assessment.
After major change	Re-validate ports, permissions, secrets, backup scope, monitoring, and rollback plan.

Change discipline

Many incidents are introduced during routine change: a firewall opened temporarily and never closed, a test credential reused in production, an old admin account left active, or a backup path broken after a database move. Treat infrastructure and security changes as documented changes with an owner, validation step, and rollback path.

PART

Evaluate Open VTS

The final step is to apply the same evidence-based security questions to Open VTS rather than treating product positioning as proof.

20 - Open VTS Data Control & Security Architecture

Open VTS is designed as customer-controlled, self-hosted vehicle tracking infrastructure. The architecture separates device ingestion, backend/application services, live/queue services, persistent

data, APIs/realtime distribution, and user applications. Security controls therefore exist across both the product and the customer-operated infrastructure boundary.

OPEN VTS DATA CONTROL & SECURITY ARCHITECTURE

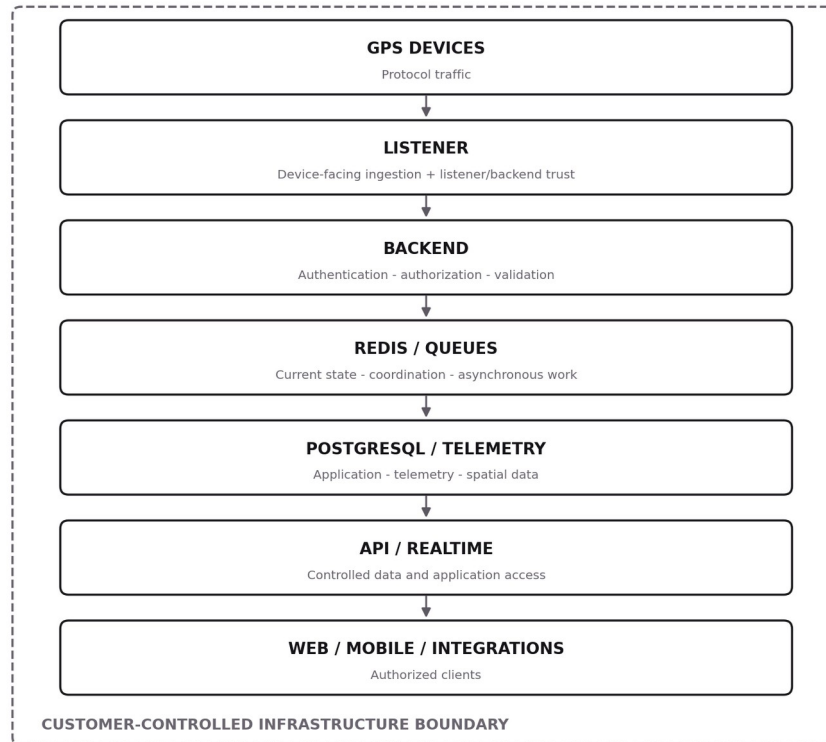


Figure 14 - Conceptual Open VTS data-control and security architecture.

Product-level areas to verify

- Authentication and token/session behavior
- Role, capability, scope, and backend authorization
- Listener-to-backend authentication
- Application request validation
- Third-party integration and workflow-secret handling
- Authorization/operational logging
- Retention mechanisms for operational datasets
- Health and observability mechanisms
- Current update and support processes

Infrastructure areas to verify

- Windows, Linux, or Docker host hardening as applicable
- Public protocol ports and HTTPS exposure
- Database/Redis isolation
- RDP/SSH or cloud administration restrictions
- DNS/TLS certificate ownership and renewal
- Backup destinations and restore procedures
- Monitoring and incident ownership

21 - What Open VTS Gives You - and What It Does Not

Open VTS provides / enables	Open VTS does not automatically provide
Customer-controlled self-hosted deployment architecture	A securely configured operating system, firewall, or cloud account.
Application authentication and authorization mechanisms	Correct permission decisions for every employee/customer.

Open VTS provides / enables	Open VTS does not automatically provide
Structured application/API access to fleet and operational data	Automatic portability of every record into any third-party platform.
Data-retention and operational-data mechanisms	A universal retention policy that fits every jurisdiction or customer contract.
Health, logging, queue/worker, and reliability mechanisms	A complete external monitoring/incident-response program without operator setup.
Software updates and product support according to current terms	Automatic deployment of infrastructure patches, backups, or TLS renewals.
Proprietary core platform deployed in customer-controlled infrastructure	Ownership of the Open VTS core source code.
Open-source mobile application	Open-source licensing for the core server platform.

This distinction is central to the Open VTS security position: control should make responsibilities clearer, not disappear them.

22 - Security Questions Before Production

Use these questions during technical acceptance. Attach evidence and an owner to every answer that matters.

1. Where do application records, telemetry, live state, logs, and backups reside?
2. Which external services receive fleet, user, or operational information?
3. Who has server/cloud, database, DNS, backup, and Superadmin access?
4. Are named privileged accounts used instead of shared credentials?
5. Is MFA enabled on hosting/cloud, registrar/DNS, and other critical control planes where supported?
6. Which roles can access which vehicles, customers, financial records, commands, and integrations?
7. Are restricted resources blocked at the backend/API boundary?
8. Which public ports are required for HTTPS and GPS protocols?
9. Are PostgreSQL, Redis, and internal services kept private?
10. How is RDP/SSH or other server administration restricted?
11. How is TLS configured and renewed for web/API traffic?
12. Which GPS device protocols do or do not provide encrypted transport?
13. How are password hashes, tokens, and reset flows protected?
14. How are listener/backend and integration secrets stored?
15. Can secrets appear in logs, workflow traces, screenshots, or exports?
16. Who can rotate or revoke production credentials?
17. What retention period applies to telemetry, events, logs, journals, and execution histories?
18. Who approves retention changes?
19. How are deleted records handled across backups and external integrations?
20. Which systems and configuration must be backed up?
21. What are the deployment RPO and RTO?
22. When was the last successful restore test?
23. Is at least one recovery copy independent from the primary server?
24. Which login, authorization, infrastructure, queue, database, and telemetry-health signals are monitored?
25. Who receives critical alerts and owns incident response?
26. What happens to incoming telemetry during selected downstream failures?
27. What happens if the deployment must be rebuilt on a new host?
28. Which functions depend on current license validation or external services?
29. How are application and operating-system updates reviewed and deployed?
30. What data and device reconfiguration would be required to exit or migrate the platform?
31. Which security requirements are contractual, regulatory, customer-driven, or internal policy?
32. Which important security limitation is accepted today, and who owns the remediation plan?

PART

Final Security & Data Control Checklist

A concise production gate: if a critical item has no owner or evidence, the deployment is not ready to treat that control as complete.

Production Security & Data Control Gate

Domain	Confirm before go-live
DATA	Critical data classes, storage locations, external flows, and owners are documented.
ACCESS	Named privileged users, role boundaries, scoped access, offboarding, and sensitive actions are verified.
NETWORK	Public ports are intentional; databases/Redis/internal services are private; administration access is restricted.
APPLICATION	Authentication, backend authorization, validation, and sensitive flows are tested.
SECRETS	Production credentials are protected, inventoried, rotatable, and absent from uncontrolled logs/source.
RETENTION	Telemetry, events, journals, logs, and backup retention have approved purposes and periods.
BACKUP	Backup scope is complete; independent copies exist where required; restore has been tested.
MONITORING	Critical platform, infrastructure, access, queue, data-growth, and integration signals are visible.
CONTINUITY	RPO/RTO, rebuild steps, incident owners, and failure/recovery procedures are documented.
EXIT	Data retrieval, tracker reconfiguration, integration rebuilding, and commercial termination conditions are understood.

Final governance question

For every critical control, can we name the owner, the implementation, the evidence, and the remaining risk?

If one of those four elements is missing, the control is not yet fully governed. This is a more useful standard than asking whether a platform is simply “secure.”

Companion Resource - Fleet Data & Security Responsibility Matrix

The recommended companion worksheet should turn the handbook into an operational register. Each row represents a control; columns record whether responsibility belongs to Open VTS, the customer, or both, along with evidence, owner, status, review date, and residual risk.

Control	Responsibility	Evidence	Owner	Review
Backend authorization	Open VTS + customer configuration	Role/scope test	Application owner	At release / access change
Server firewall	Customer	Firewall rule review	Infrastructure owner	Monthly / change
TLS certificate	Shared	Valid certificate + renewal test	Infrastructure owner	Before expiry
Database backup	Customer	Backup log + restore test	DB/infra owner	Daily + quarterly restore
Protocol security limitation	Shared / hardware-dependent	Device/protocol review	Telematics engineering	At device onboarding
Retention policy	Customer with product mechanisms	Approved policy + cleanup evidence	Data owner	Quarterly

Glossary

Term	Meaning
Access control	Rules that determine which identities may perform which actions on which resources.
Audit trail	Evidence intended to show who performed or attempted a relevant action and when.
Backup	A recoverable copy of data/configuration kept to support restoration after loss or corruption.
Custody	The physical or logical holding of data by an infrastructure or service provider.
Data minimization	Collecting and retaining only information justified by a defined purpose.
Data portability	The practical ability to extract and move useful data to another environment or platform.
Encryption at rest	Protection applied to stored data or storage media; implementation details vary by service.
Encryption in transit	Protection of data while moving between systems, commonly using TLS for web/API traffic.
Least privilege	Granting only the access required to perform a legitimate responsibility.
MFA	Multi-factor authentication; use of more than one factor to verify an identity.
RPO	Recovery Point Objective - the maximum acceptable amount of recent data loss.
RTO	Recovery Time Objective - the maximum acceptable period before service is restored.
Retention	The approved period for which a data class is kept.
Secret	A sensitive credential or token that grants access to a system, service, integration, or cryptographic function.
Self-hosted	Software deployed in infrastructure selected or controlled by the organization using it.
Scope	The resource or organizational boundary within which a permission applies.
Shared responsibility	A model in which software-provider and customer/infrastructure responsibilities are explicitly divided.
Telemetry	Time-stamped operational data reported by a device, such as location, speed, ignition, or sensors.
TLS	Transport Layer Security - a protocol used to protect network communication such as HTTPS.
Zero trust	A security approach that avoids implicit trust and continuously evaluates identity, device, and access context; not a product checkbox.

References & Further Reading

Use current editions of authoritative sources when translating this handbook into organization-specific policy. Relevant references include:

- NIST Cybersecurity Framework (CSF) 2.0 - National Institute of Standards and Technology.
- NIST SP 800-53 - Security and Privacy Controls for Information Systems and Organizations.
- NIST SP 800-61 - Computer Security Incident Handling guidance (use current NIST incident-response guidance at publication time).
- OWASP Application Security Verification Standard (ASVS).
- OWASP API Security Top 10.
- CIS Critical Security Controls.
- ISO/IEC 27001 - Information security management systems.
- PostgreSQL official security, authentication, backup, and recovery documentation.
- Redis official security guidance.
- Current operating-system, firewall, cloud, Docker/container, and TLS documentation applicable to the chosen deployment environment.

- Device-manufacturer protocol specifications when evaluating transport security and device authentication behavior.

Open VTS resources

- Open VTS Security resources
- Open VTS Deployment & Infrastructure Planning Guide
- Open VTS API Reference
- Open VTS Supported Devices & Ports
- Open VTS documentation and current release notes
- Current Open VTS license, pricing, update entitlement, and support terms
- Open VTS mobile application source/resources

Before final publication or enterprise use, verify release-specific technical claims, supported deployment behavior, licensing dependencies, retention configuration, and security controls against the current Open VTS release.

About Open VTS

Open VTS is a customer-controlled, self-hosted vehicle tracking platform for GPS and telematics businesses, fleet operators, integrators, and organizations that want greater authority over the infrastructure and operational data environment behind their tracking system. The platform brings together device-protocol connectivity, telemetry processing, data services, live fleet operations, reports, alerts, roles and permissions, APIs, realtime communication, workflow automation, and web and mobile experiences.

The core Open VTS platform is proprietary software deployed using supported self-hosted environments. The Open VTS mobile application is separately available with source code. Open VTS is designed to reduce software deployment complexity while preserving the responsibilities and controls required to operate serious tracking infrastructure.

Your Infrastructure. Your Fleet Data. Your Rules.

Open VTS | Track Without Limits. Powered By You.

Open VTS Guide Series

Guide	Strategic job
01 - The Self-Hosted GPS Tracking Buyer's Guide	Evaluate
02 - The Business Case for Self-Hosted GPS Tracking	Justify
03 - The GPS Platform Migration Playbook	Migrate
04 - Open VTS Deployment & Infrastructure Planning Guide	Deploy
05 - The Architecture of Modern GPS Tracking	Understand
06 - Fleet Data Ownership & Security Handbook	Trust

Publication integrity

Open VTS guides are designed to teach the category before promoting the product. Security, performance, licensing, deployment, and device-support claims should be tied to current evidence. When product behavior, infrastructure requirements, or commercial terms change, the publication should be reviewed and versioned rather than allowing an old statement to become permanent marketing debt.

Official resource destination: openvts.io | Guide 06 of 06 | 2026 Edition | Version 1.0