LogIQ
VDA5050 Analyzer

PRODUCT BROCHURE

# Prove conformance. Don't assume it.

A conformance checker for VDA5050, M2X and VDMA LIF — for log files and for live traffic.

**112**
checks for VDA5050 and M2X

**13**
checks for VDMA LIF

**2.1.0 / 3.0**
VDA5050 versions supported

A PRODUCT BY

LogistiXpert

**www.logistixpert.de**
vda5050.logistixpert.de · m.merz@logistixpert.de

# At a glance

The LogIQ VDA5050 Analyzer reads the traffic between a fleet controller and its vehicles, checks it against the specification and states verifiably what deviates — from a log file just as from live operation.

### Three specifications

VDA5050 2.1.0 and 3.0, M2X 0.9.0 as a preview and VDMA LIF 1.0.0 in one tool.

### Three ways in

Import a log file, record live from the broker or sit in between as a transparent proxy.

### A verdict you can check

The report states not only the findings but the scope: what ran, what found no data and what was switched off.

### Fleet-ready

Sequences and lifecycles are evaluated per vehicle — the way VDA5050 counts them.

### See it, don't read it

A live view on the real track, replay of finished runs, the order path over the LIF layout.

### In your own network

Docker Compose, on-premise or hosted. Broker credentials encrypted, roles with clear limits.

## Contents

# 01 Why this tool exists

VDA5050 promises that vehicles and fleet controllers from different vendors work together. Whether they do is decided during commissioning — and that is usually where the one thing both sides could agree on is missing: a shared, checkable view of the messages that actually crossed the broker.

### “Who is right?”

Vehicle and controller come from different vendors. When something breaks it is one claim against another. Without evidence the argument takes days and often ends with whoever gives in first.

### “What was actually checked?”

A green tick without a scope is worthless. An acceptance test has to state which rules ran, which found no matching messages and which were deliberately switched off.

### “What happened there?”

The fault appears once, on a Friday evening. Without a recording that keeps its context, all that is left is to reproduce it — and hope that it shows.

### The approach

The standard describes messages. It does not prove that two systems read them the same way. That is the gap the analyzer closes: it evaluates the real traffic, anchors every finding in the message that triggered it and documents the rules the verdict was reached under.

### What the analyzer is not

It replaces no safety assessment, no acceptance drive and no test house. It issues no certification and no release approval. What it delivers is evidence: which message was sent when and by whom, which rule applies to it and exactly where that rule is broken. Judging that evidence remains yours.

# 02 Three ways in

Whichever you have — a file from the field, a live broker or an installation nobody is allowed to touch.

### 1 Import a log file

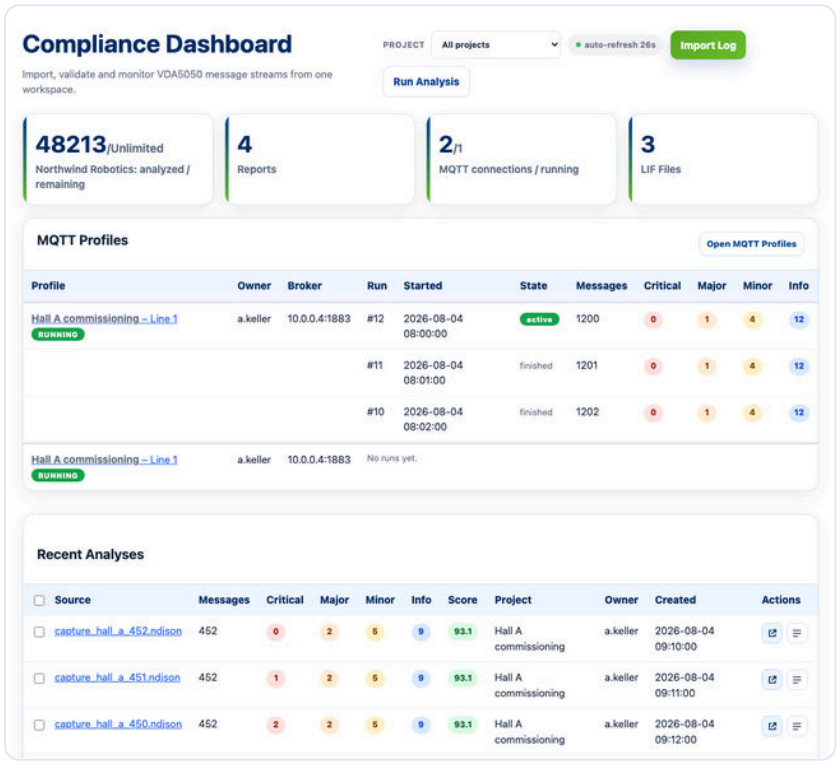
Recordings as .log, .txt, .json or .ndjson. The importer detects the message type, reports lines it could not read and keeps the field order of the source — message views therefore match your file character for character. Cleaned JSON, NDJSON and the untouched original stay available for download.

### 2 Record live

A capture profile holds broker, credentials, topics, target version, check selection and optionally a LIF layout. Started profiles record in the background and analyse every message as it arrives; several profiles run at once. Every start-stop cycle becomes a run with its own messages and findings.

### 3 Sit in between

The same profile can mirror the recorded traffic onto a second broker. The analyzer then runs as a transparent proxy between vehicles and controller without changing operation — useful when only one network segment is reachable.




One workspace for imports, analyses, layouts and running captures.

> An import needs no access to the installation at all. A live capture needs read access to the broker; mirroring is optional and enabled per profile.

# 03 How it checks

Checking is a chain: a message first has to match the schema, then its structure has to hold, then its sequence and only then can order and state be judged against each other.

### Schema

Every message against the official JSON schema of the target version. 2.1.0 and 3.0 are kept apart because 3.0 renamed fields — a tool that knows only one version reports errors that are none as soon as the version changes.

### Structure

Required fields, data types, ISO-8601 timestamps in UTC, semantic version, collections as arrays. Enumerations are checked against the values the target version allows.

### Sequence

headerId gaps and regressions, timestamp regressions, node and edge sequences, action lifecycles from issue to terminal state, battery and safety reporting.

### Coherence

Is an issued order acknowledged? Does the reported progress match it? Does lastNodeSequenceId stay monotonic within the order? This correlation runs per vehicle.

## Severities

| | |
|---|---|
| Critical | The message does not conform; the other side may reject it. |
| Major | A clear deviation that affects the sequence of operations. |
| Minor | A deviation without immediate effect but worth reporting. |
| Info | Context, not a defect. |

### A fleet is not one vehicle times ten

VDA5050 counts headerIds per topic and the topic carries manufacturer and serial number. That is exactly how the analyzer evaluates: every sequence, every lifecycle, every order-state correlation, per vehicle. Laid out in one row, two vehicles that both start at headerId 1 would look like a single sender jumping backwards — hundreds of findings that are none. Every finding also carries “manufacturer/serial” and can be filtered by it.

### Score and verdict

The findings produce a score per area, an overall score and a verdict (PASS, WARNING, FAIL). Both are worth only as much as the scope printed next to them — hence the next page.

# 04 Declaration against reality

Every vehicle declares in its factsheet what it can do. 22 checks hold the traffic against that declaration, and they answer a different question from the rest of the ruleset: not “does this message follow the standard” but “does it suit this vehicle”.

### Actions

Every actionType in an order or an instant action has to appear in the vehicle's catalogue — in the right scope, with a declared blockingType and with the parameters the vehicle expects.

### Optional parameters

The specification is sharp here: what is not listed counts as not supported. The analyzer reports optional fields that are sent although the vehicle never claimed to support them.

### Limits

Message length, length and character set of identifiers, nodes and edges per order, actions per node, nodeStates and actionStates per state — both directions are measured against the declared limits.

### Timing

Minimum interval between orders, minimum and default interval between states. Judged with a tolerance set per analysis — a percentage band plus a fixed slack, because the clocks of the systems involved are not synchronised.

### Whose fault was it?

Every finding in this group names the side that caused it: the fleet manager asking for something undeclared, or the vehicle failing to keep its own word. In the report that is a filter. The most common commissioning argument therefore ends in an attribution rather than in assertions.

### No factsheet, no statement

Without a factsheet, or where a factsheet declares nothing for an area, the affected check reports “no data” — never “passed”. A report claiming conformance against a reference nobody supplied would be exactly the false confidence this tool exists to prevent.

### Three ways to a factsheet

From the traffic, when the recording contains the factsheet topic. Imported as a file of its own — log exports from a fleet manager rarely carry that topic. Or chosen when setting up the analysis; that choice wins over the factsheet in the traffic, and the report records which one applied. An imported factsheet is checked on the way in and can be read formatted in the tool — it becomes the reference for 22 checks, so it is not taken on trust.

# 05 Report and auditability

A test report that only lists faults is no basis for an acceptance test. The analyzer puts the how next to the what.

### Active

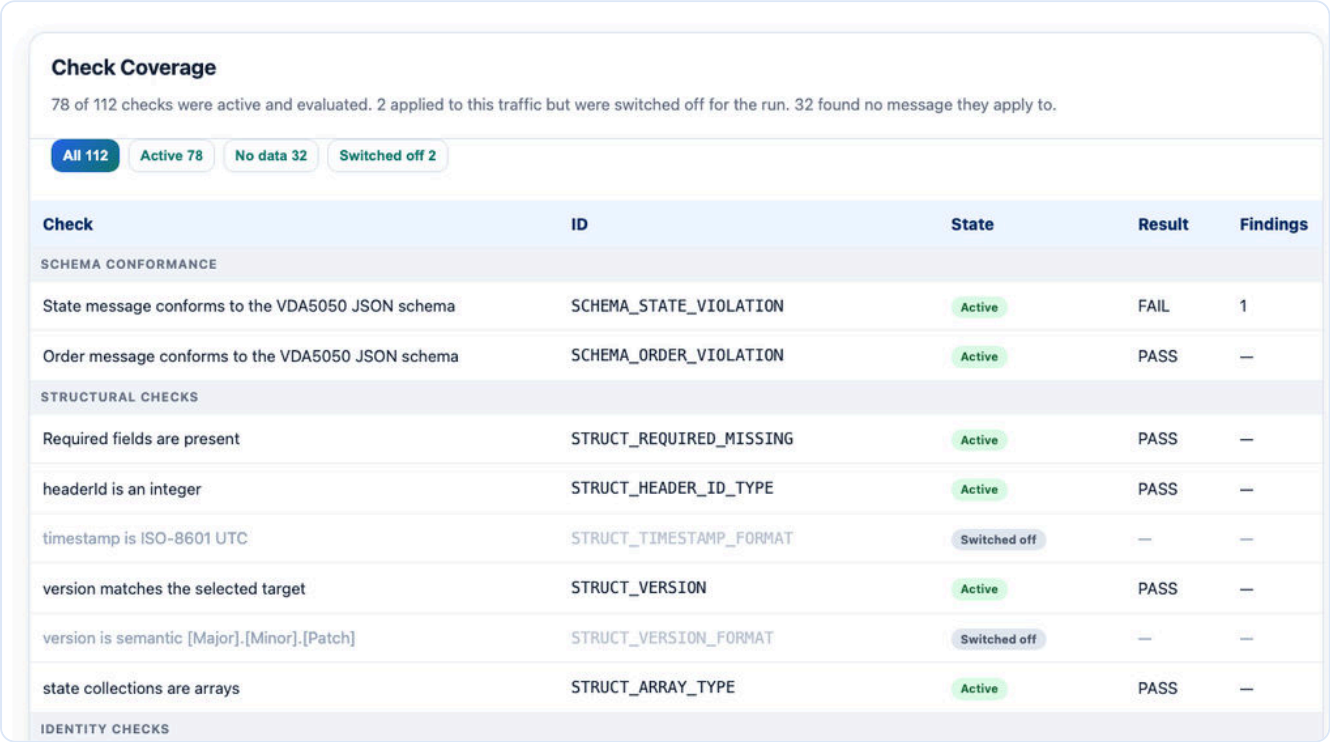
The rule ran and judged — PASS or FAIL, with the number of findings.

### No data

The rule ran but found no message it applies to. It does not pass silently; it is reported as having no data.

### Switched off

Deselected on purpose — and visible in the report rather than hidden.



**Check Coverage**

78 of 112 checks were active and evaluated. 2 applied to this traffic but were switched off for the run. 32 found no message they apply to.

All 112 | Active 78 | No data 32 | Switched off 2

| Check | ID | State | Result | Findings |
|---|---|---|---|---|
| SCHEMA CONFORMANCE | | | | |
| State message conforms to the VDA5050 JSON schema | SCHEMA_STATE_VIOLATION | Active | FAIL | 1 |
| Order message conforms to the VDA5050 JSON schema | SCHEMA_ORDER_VIOLATION | Active | PASS | — |
| STRUCTURAL CHECKS | | | | |
| Required fields are present | STRUCT_REQUIRED_MISSING | Active | PASS | — |
| headerId is an integer | STRUCT_HEADER_ID_TYPE | Active | PASS | — |
| timestamp is ISO-8601 UTC | STRUCT_TIMESTAMP_FORMAT | Switched off | — | — |
| version matches the selected target | STRUCT_VERSION | Active | PASS | — |
| version is semantic [Major].[Minor].[Patch] | STRUCT_VERSION_FORMAT | Switched off | — | — |
| state collections are arrays | STRUCT_ARRAY_TYPE | Active | PASS | — |
| IDENTITY CHECKS | | | | |

Scope of one run: 78 of 112 active, 32 without matching data, 2 switched off.

### Audit trail

Every run records the tool and ruleset version, the target version of the specification, the schema revisions used, the number of input messages, a checksum over the input and the counts of checks total / enabled / evaluated. That makes a result repeatable and a report verifiable.

### Output

Every report can be downloaded as JSON and as Markdown — for the project archive, as an annex to an acceptance test or for further processing.

# 06 Findings in context

A finding is only useful if it can be traced. Every entry therefore leads to the message that triggered it.

### The place

The finding names the check, the severity, the field path and the message it occurred in — and shows that message with the field highlighted, not just a line number.

### The context

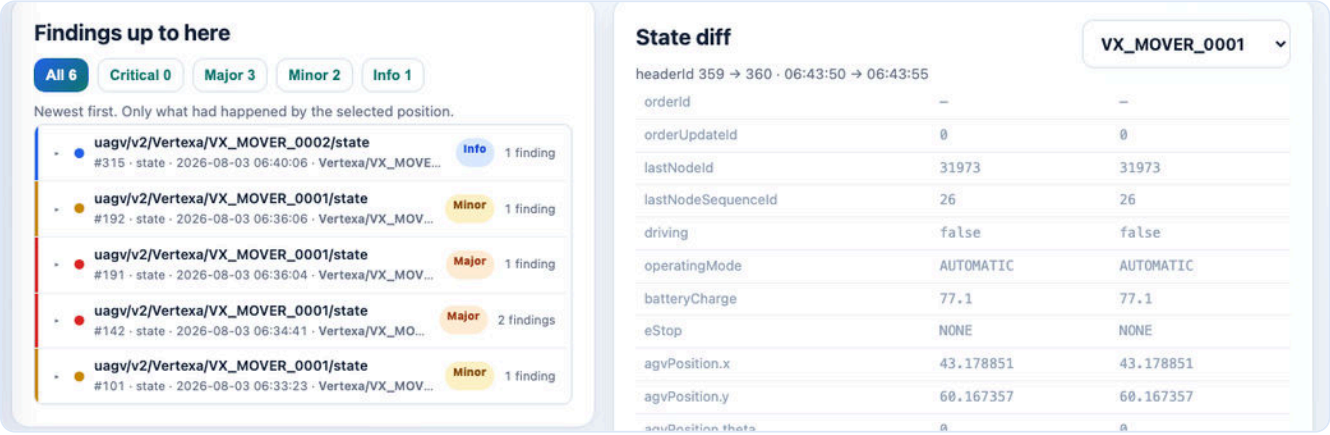
Previous, current and next message sit side by side and scroll as one, switchable between the vehicle's own stream — previous and next message of the same kind from the same vehicle — and the log, where the neighbours often belong to another robot.

### The view

Filter by severity and by vehicle; group by check or by message — depending on whether you are after a pattern or a single incident.

### The comparison

Two consecutive state messages from the same vehicle, field by field, with the changes highlighted. “What changed” is answered without comparing JSON by hand.




Findings grouped by message, next to the field-by-field comparison.

# 07 Layout and visualization

Orders refer to a track. When order and layout do not match the vehicle rejects the order — and without a picture that takes a long time to find.

1

### Check the LIF

13 rules from VDMA LIF 1.0.0: schema, node and edge integrity, reachability across the network, station references. The result is a report of its own with a drawing of the layout.

2

### Order over layout

The path of an order is drawn over the imported layout. The comparison reports node by node and edge by edge whether the order fits and on a mismatch names every missing node and edge.

3

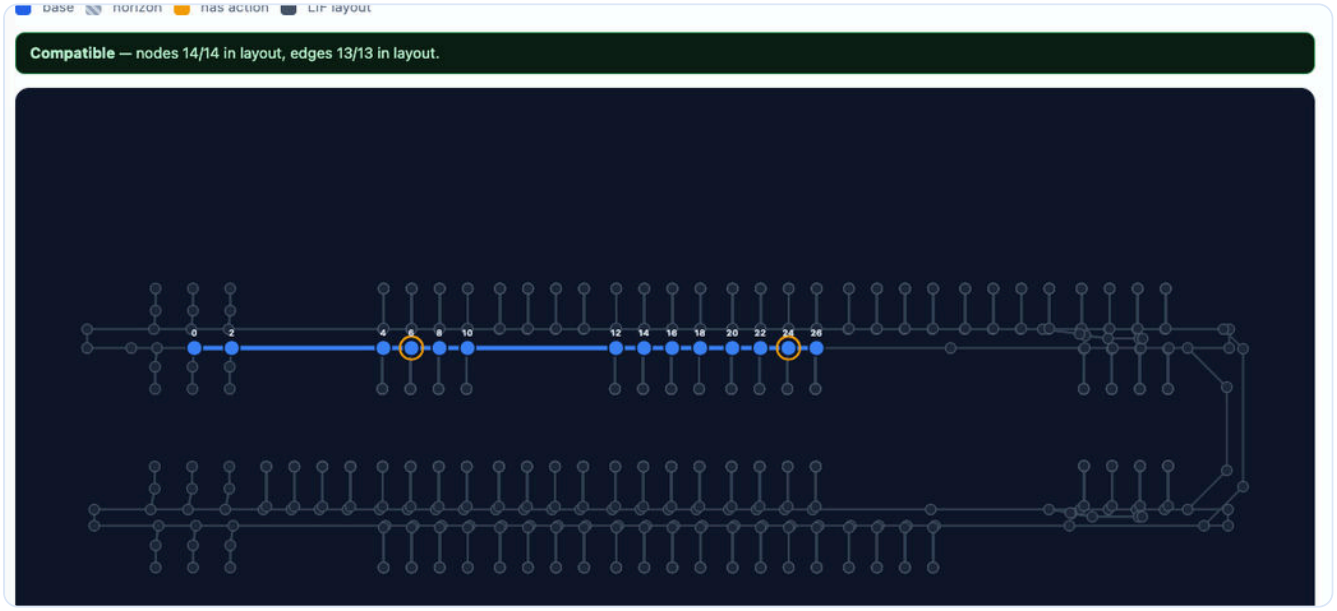
### State over time

Next to the order path, the visualization shows selected state fields over time — position, battery or operating mode, for instance.

4

### Stays with the report

The layout is chosen when the analysis is set up and stays attached to the report. Whoever opens it later sees the same basis.




Order path over the LIF layout with the comparison result.

# 08 Live view

While a profile records, you can watch the installation at work — on the same layout the reports use.

- Every vehicle on the LIF layout, each in its own colour.
- The last order issued as a path, the last reported position as a marker.
- A vehicle table with order, last node, position, battery, state and the age of the report.
- Findings appear as they happen, filterable by severity and vehicle.
- The state diff of a vehicle chosen from a dropdown next to it.
- Refreshes every second; the page pauses while it is in the background.


Two vehicles with their own routes and last known positions.

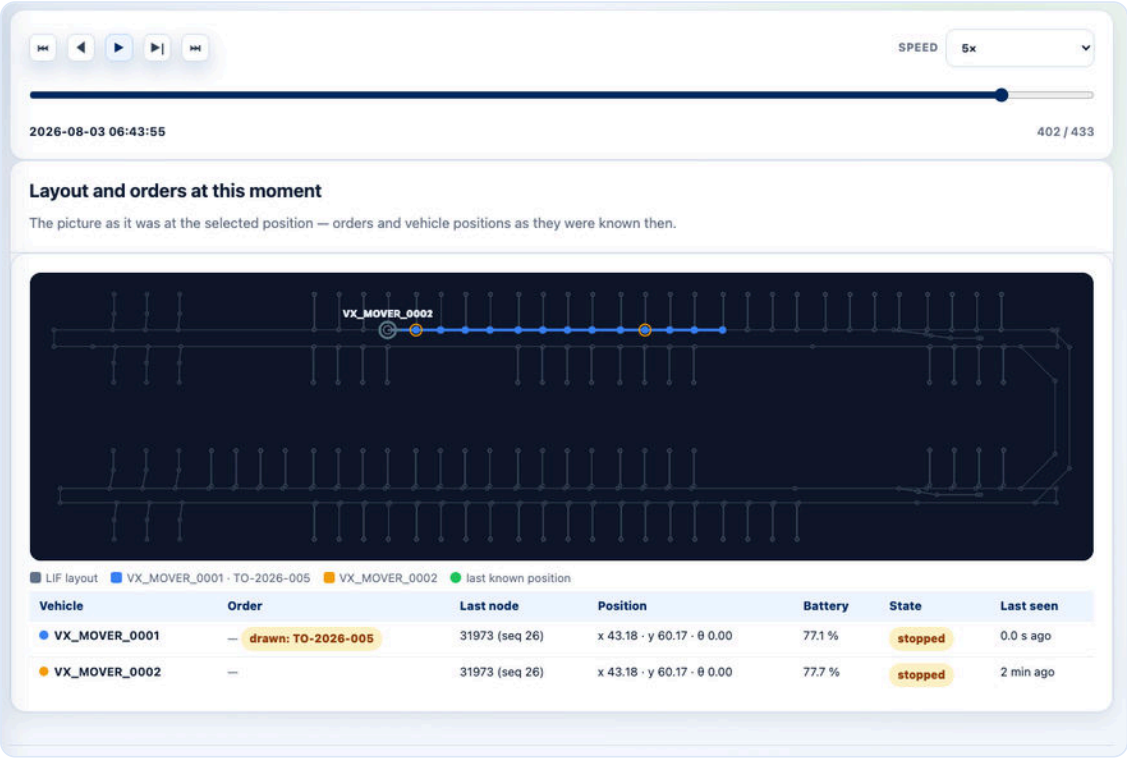
### The actual point

The map draws the order last sent. The table shows what the vehicle itself reports. When the two differ, it says so — and that gap is where most commissioning faults live.

# 09 Replay

The same view for everything that already happened: finished captures and imported log files can be played back.

- Forwards, backwards, step by step, up to twenty times speed.
- The gaps are the real message intervals, capped so that long pauses do not hold up a demonstration.
- The finding list shows only what was already known at the selected point — no knowledge from the future.
- The overlaid LIF layout can be switched without changing the report.




Replay at position 402 of 433 with the findings known by then.

### What it is for

The post-mortem. An incident can be played back in front of the supplier instead of described; everyone in the room sees the same sequence, with the same findings in the same places.

# 10 Unattended operation and reporting

A recording over a weekend brings a hundred thousand messages. Nobody reads those — so the analyzer reports by itself.

1. ### Report on a schedule

   Every X hours, a digest of what turned up since the last one. A quiet period is reported too, so that silence stays unambiguous.

2. ### Alert on a threshold

   Or only once enough has piled up: after X findings, counting Critical only, Critical and Major or every severity.

3. ### Recipients per profile

   The addresses belong to the capture profile. Different installations can have different distribution lists.

4. ### Resumable

   A restart picks the analysis up where it stopped. Nothing is counted twice, nothing is skipped.

> ### What the e-mail says
>
> Period, trigger and the number of new findings, broken down by severity, by vehicle and by check — with a direct link into the running capture.


Digest of a threshold alert.

# 11 Teams, projects and contingents

Test results do not belong on one laptop. The analyzer knows groups, projects and clear rights.

1. ### Enterprise

   Every member sees what the group produced; every list names the owner. Shared visibility is not a shared delete right, though: only superusers may delete a colleague's data.

2. ### Superuser

   Manages membership and projects and may delete a colleague's data. Deleting accounts stays with the administrator.

3. ### Fallback superuser

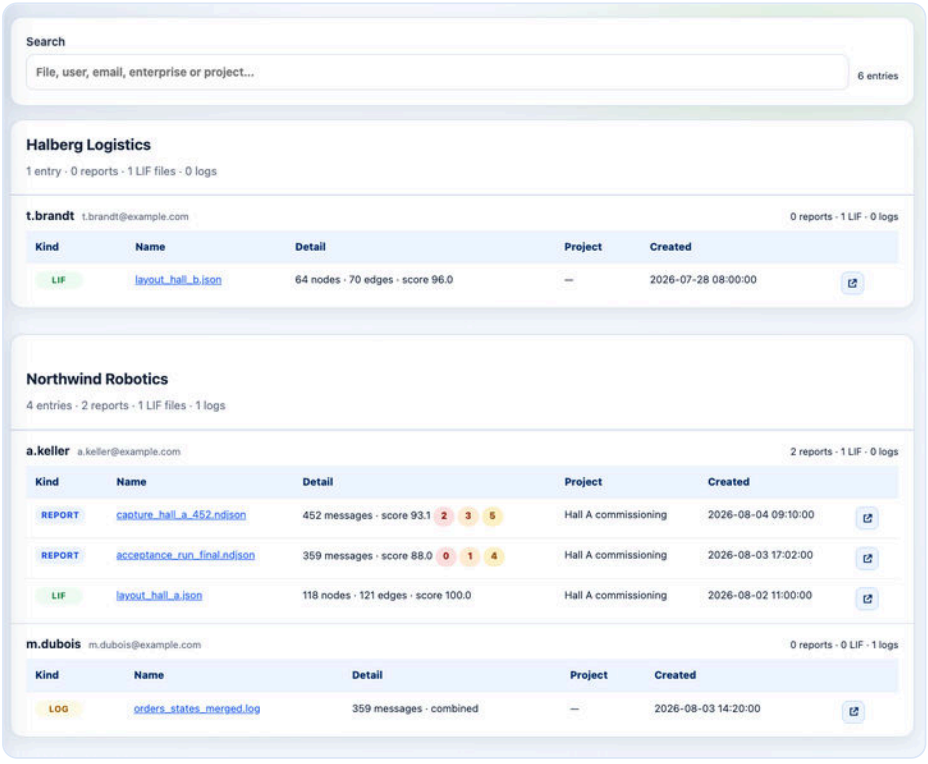
   Exactly one per enterprise. When a member closes their account, their logs, layouts, reports and captures pass to that person instead of being lost with the account.

4. ### Projects

   Logs, layouts, profiles and reports can be filed under a job and filtered by it. A completed project drops out of the lists without being deleted and comes back when it is reopened.

5. ### Contingents

   Analysed messages are counted. An enterprise draws on one shared contingent, individual users on their own. What is counted is consumption, not stock — a deleted report does not return quota.




The administrator's view across enterprises, users and artefacts, with a full-text search.

# 12 Operations and security

Recordings from an installation are sensitive. The analyzer is built so that they never have to leave the plant.

### On your own premises

Docker Compose, one file, one command. On your hardware, in your network or as a hosted instance — the choice changes nothing about what the tool does.

### Encrypted connection

MQTT over TLS, with your own CA certificate where the site uses one. The proxy target has the same switch, so a mirrored connection is never weaker than the one it mirrors.

### Credentials encrypted

Broker passwords are stored encrypted, not in clear text. They stay usable for reconnects but are not readable.

### A licence that does not call home

On your own network the installation carries a signed licence file, checked locally. Fourteen days of grace on expiry, then read-only — everything recorded stays reachable.

### Reports under your own name

With an Enterprise licence the report, the export and the e-mail digest carry the customer's name and mark. What the tool is stays written underneath.

### Links that cannot be guessed

Reports, runs, imports and layouts hang off unguessable ids. On top of that, every access checks ownership.

### Roles with limits

Member, superuser, administrator. Who may delete what is defined in one place in the code and enforced on every action.

### Closing an account

Users can delete their own account. Without an enterprise everything goes with it; inside a group the work stays with the group and passes to the fallback superuser.

### Quality on record

654 automated tests run before every release, among them regression tests against fixed sample data.

### Data and retention

Uploads, cleaned messages, reports and recordings live in one data directory on the instance. Deletion goes through the application — one entry at a time or a whole selection. Deleting a log keeps the reports made from it, because a report is the record of a test run and outlives its source. For quota the opposite holds: messages once analysed stay counted.



### Note

# 13 Technical data

| | |
|---|---|
| **Specifications** | VDA5050 2.1.0 and 3.0 · M2X 0.9.0 (preview) · VDMA LIF 1.0.0 |
| **Checks** | 112 for VDA5050 and M2X, 13 for LIF; individually selectable, per analysis and per capture profile |
| **Message types** | order, state, connection, instantActions, factsheet, visualization; M2X: load handling, door, lift, signal light, charging device, transport order |
| **Input formats** | .log, .txt, .json, .ndjson — raw recordings, MQTT captures and NDJSON exports |
| **Output** | Report as HTML, JSON and Markdown; recordings exportable as NDJSON |
| **Live operation** | MQTT 3.1.1/5 via paho-mqtt, plain or over TLS (port 8883, your own CA if you have one), user and password, several profiles at once, optional mirroring onto a second broker |
| **Notification** | E-mail over SMTP, on a schedule or a threshold, recipients per profile |
| **Deployment** | Docker Compose (application and broker), SQLite for storage, on-premise or hosted |
| **Use** | Web application, nothing to install on the workstation; interface in English |
| **Storage** | Uploads, cleaned messages, reports and recordings live in one data directory; deletion goes through the application |

## Requirements

- A machine or VM with Docker; enough disk space for continuous recordings.
- Read access to the installation's MQTT broker, plus write access to the target broker for mirroring.
- An SMTP account for e-mail reports; without one, mails are written to a file.

## Limits and assumptions

- The analyzer judges what it sees. Messages that never reach the broker cannot be assessed.
- A replay covers up to 20,000 messages; longer runs are truncated and the truncation is stated in the tool.
- M2X follows a pre-release of the specification; results are marked as a preview.
- Attributing findings to a vehicle requires manufacturer and serial number in the topic or the message.

# 14 Check catalogue

Every check the tool performs, grouped by area. Identifiers and short texts are the ones used in the application and in the report; there, each check additionally carries its reference into the specification.

## VDA5050 and M2X (112)

### Schema Conformance (2)

`SCHEMA_STATE_VIOLATION`
State message conforms to the VDA5050 JSON schema

`SCHEMA_ORDER_VIOLATION`
Order message conforms to the VDA5050 JSON schema

### Structural Checks (6)

`STRUCT_REQUIRED_MISSING`
Required fields are present

`STRUCT_HEADER_ID_TYPE`
headerId is an integer

`STRUCT_TIMESTAMP_FORMAT`
timestamp is ISO-8601 UTC

`STRUCT_VERSION`
version matches the selected target

`STRUCT_VERSION_FORMAT`
version is semantic [Major].[Minor].[Patch]

`STRUCT_ARRAY_TYPE`
state collections are arrays

### Identity Checks (2)

`IDENTITY_MANUFACTURER_CHANGED`
Report streams covering more than one manufacturer

`IDENTITY_SERIAL_CHANGED`
Report streams covering more than one vehicle

### Header / Time Checks (4)

`HEADER_DUPLICATE`
No duplicate headerId values

`HEADER_REGRESSION`
headerId increases monotonically

`HEADER_GAP`
Detect headerId gaps

`TIMESTAMP_REGRESSION`
timestamp increases or stays equal

### Node / Edge Checks (7)

`STATE_ENTRY_TYPE`
nodeStates/edgeStates entries are objects

`STATE_ENTRY_INCOMPLETE`
nodeId/edgeId and sequenceId are present

`STATE_SEQUENCE_TYPE`
sequenceId is an integer

`STATE_SEQUENCE_REGRESSION`
sequenceIds increase within a message

`NODE_EDGE_SEQUENCE_OVERLAP`
Report node/edge sequenceId overlaps

`LAST_NODE_REGRESSION`
lastNodeSequenceId does not decrease per orderId

`LAST_NODE_JUMP`
lastNodeSequenceId does not jump by more than 4

### Action Lifecycle Checks (5)

`ACTION_ENTRY_TYPE`
actionStates entries are objects

`ACTION_REQUIRED_MISSING`
actionId and actionStatus are present

`ACTION_STATUS_INVALID`
actionStatus uses an allowed enum value

`ACTION_INVALID_TRANSITION`
Action transitions are allowed

`ACTION_NOT_TERMINAL`
Actions reach a terminal state

### Operating Mode Checks (1)

`STATE_OPERATING_MODE_INVALID`
operatingMode uses an allowed enum value

### Driving / Velocity Checks (4)

`PAUSED_AND_DRIVING`
paused=true and driving=true are not both active

`VELOCITY_MISSING_WHILE_DRIVING`
Velocity is present when driving=true

`DRIVING_ZERO_VELOCITY`
Report driving=true with zero velocity

`NOT_DRIVING_WITH_VELOCITY`
Report driving=false with velocity

### Battery Checks (2)

`BATTERY_CHARGE_RANGE`
battery charge is between 0 and 100

`BATTERY_CHARGE_JUMP`
Detect battery charge jumps

### Safety Checks (4)

`SAFETY_FIELD_MISSING`
safetyState contains the eStop and fieldViolation fields

`SAFETY_ESTOP_VALUE_INVALID`
emergency-stop value uses an allowed enum

`ESTOP_WHILE_DRIVING`
Report active eStop with driving=true

`FIELD_VIOLATION_WHILE_DRIVING`
Report fieldViolation=true with driving=true

### Error Reporting Checks (1)

`STATE_ERROR_STRUCTURE`
errors contain errorType and a valid errorLevel

### Order Structural Checks (5)

`ORDER_REQUIRED_MISSING`
Required order fields are present

`ORDER_HEADER_ID_TYPE`
Order headerId is an integer

`ORDER_TIMESTAMP_FORMAT`
Order timestamp is ISO-8601 readable

`ORDER_VERSION`
Order version matches the selected target

`ORDER_ARRAY_TYPE`
nodes and edges are arrays

### Order Update Checks (3)

`ORDER_UPDATE_DUPLICATE`
No duplicate orderUpdateId values

`ORDER_UPDATE_REGRESSION`
orderUpdateId increases monotonically

`ORDER_UPDATE_GAP`
Detect orderUpdateId gaps

## VDA5050 and M2X (continued)

### Order Graph Checks (7)

ORDER_SEQUENCE_TYPE
node and edge sequenceId values are integers

ORDER_SEQUENCE_PARITY
Node and edge sequenceId parity is valid

ORDER_SEQUENCE_ORDER
Nodes and edges are sorted by sequenceId

ORDER_DUPLICATE_ID
Node, edge and action IDs are unique

ORDER_EDGE_ENDPOINT_UNKNOWN
Edges reference existing nodes

ORDER_GRAPH_CHAIN
Graph sequence is continuous

ORDER_RELEASED_CONTIGUITY
Released base is contiguous

### Order Position Checks (1)

ORDER_POSITION_MISSING
Node positions contain coordinates and mapId

### Order Action Checks (2)

ORDER_ACTION_REQUIRED_MISSING
Order actions contain required fields

ORDER_BLOCKING_TYPE
blockingType uses allowed values

### Order Trajectory Checks (1)

ORDER_TRAJECTORY_STRUCTURE
Edge trajectory structure is valid

### Connection Checks (4)

CONN_REQUIRED_MISSING
Required connection fields are present

CONN_HEADER_ID_TYPE
Connection headerId is an integer

CONN_VERSION
Connection version matches the selected target

CONN_STATE_INVALID
connectionState uses an allowed enum value

### Instant Action Checks (6)

IA_REQUIRED_MISSING
Required instantActions fields are present

IA_HEADER_ID_TYPE
instantActions headerId is an integer

IA_VERSION
instantActions version matches the selected target

IA_ACTION_REQUIRED_MISSING
Instant actions contain required fields

IA_BLOCKING_TYPE
Instant action blockingType uses allowed values

IA_DUPLICATE_ACTION_ID
Instant action IDs are unique

### Factsheet Checks (3)

FS_REQUIRED_MISSING
Required factsheet objects are present

FS_HEADER_ID_TYPE
Factsheet headerId is an integer

FS_VERSION
Factsheet version matches the selected target

### Mission Lifecycle Checks (8)

MISSION_STATE_UNKNOWN_ORDER
State references a known orderId

MISSION_ORDER_NOT_ACKED
State acknowledges the orderId after an order

MISSION_ORDER_ACK_SLOW
Order is acknowledged within the timeout

MISSION_NODE_NOT_IN_ORDER
nodeStates contain only nodes from the order

MISSION_EDGE_NOT_IN_ORDER
edgeStates contain only edges from the order

MISSION_ACTION_NOT_IN_ORDER
actionStates contain only actions from the order

MISSION_ORDER_UPDATE_NOT_REJECTED
Stale order updates are rejected

MISSION_INSTANT_ACTION_NOT_REFLECTED
instantActions are reflected in state

# VDA5050 and M2X (continued)

## Capability Checks (22)

CAP_ACTION_NOT_DECLARED
Action type is declared in the factsheet

CAP_ACTION_SCOPE_INVALID
Action is used within its declared scope

CAP_ACTION_BLOCKING_TYPE_INVALID
blockingType is one the vehicle declares

CAP_ACTION_PARAM_UNDECLARED
Action parameters are declared

CAP_ACTION_PARAM_REQUIRED_MISSING
Required action parameters are sent

CAP_ACTION_PARAM_TYPE_MISMATCH
Action parameter values match the declared type

CAP_ACTION_CANCEL_NOT_ALLOWED
cancelOrder respects cancelAllowed

CAP_ACTION_PAUSE_NOT_ALLOWED
startPause respects pauseAllowed

CAP_OPTIONAL_PARAM_UNSUPPORTED
Only declared optional parameters are sent

CAP_OPTIONAL_PARAM_REQUIRED_MISSING
Parameters declared REQUIRED are sent

CAP_OPTIONAL_PARAM_UNKNOWN_PATH
Declared optional parameters exist in the schema

CAP_LIMIT_MSG_LEN
Messages stay within the declared length

CAP_LIMIT_ID_LEN
Identifiers stay within the declared length

CAP_LIMIT_ID_NUMERICAL
Identifiers are numeric when declared so

CAP_LIMIT_LOAD_ID_LEN
loadId stays within the declared length

CAP_LIMIT_TOPIC_LEN
Topic elements stay within the declared length

CAP_LIMIT_ARRAY_LEN_ORDER
Orders respect the declared array limits

CAP_LIMIT_ARRAY_LEN_STATE
States respect the declared array limits

CAP_TIMING_ORDER_INTERVAL
Orders keep the declared minimum interval

CAP_TIMING_STATE_INTERVAL_MIN
States keep the declared minimum interval

CAP_TIMING_STATE_INTERVAL_DEFAULT
States arrive within the declared default interval

CAP_TIMING_VISUALIZATION_INTERVAL
Visualization keeps the declared interval

## M2X Schema Conformance (6)

M2X_SCHEMA_LHS_VIOLATION
Load handling station messages conform to the M2X JSON schema

M2X_SCHEMA_DOOR_VIOLATION
Door messages conform to the M2X JSON schema

M2X_SCHEMA_ELEVATOR_VIOLATION
Elevator messages conform to the M2X JSON schema

M2X_SCHEMA_SIGNAL_VIOLATION
Signaling device messages conform to the M2X JSON schema

M2X_SCHEMA_CHARGING_VIOLATION
Charging device messages conform to the M2X JSON schema

M2X_SCHEMA_ORDER_VIOLATION
Transport order messages conform to the M2X JSON schema

## M2X Lifecycle Checks (6)

M2X_REQUEST_NOT_REFLECTED
Requests appear in the entity state

M2X_MULTIPLE_ACTIVE_REQUESTS
Only one request is active per entity

M2X_REQUEST_INVALID_TRANSITION
Request status transitions are allowed

M2X_INTERACTION_SEQUENCE
Interaction steps follow the specified order

M2X_STATE_HEARTBEAT_MISSED
Load handling stations publish state at least every 30 seconds

M2X_REQUEST_TYPE_NOT_SUPPORTED
Requests match the addressed interface

## VDMA LIF 1.0.0 (13)

### Schema Conformance (1)

`SCHEMA_LIF_VIOLATION`
LIF file conforms to the LIF JSON schema

### LIF Structural Checks (4)

`LIF_DUPLICATE_LAYOUT_ID`
layoutId is unique

`LIF_DUPLICATE_NODE_ID`
nodeId is unique across the file

`LIF_DUPLICATE_EDGE_ID`
edgeId is unique across the file

`LIF_DUPLICATE_STATION_ID`
stationId is unique across the file

### LIF Graph Checks (5)

`LIF_EDGE_START_NODE_UNKNOWN`
Edge startNodeId is part of the same layout

`LIF_EDGE_END_NODE_UNKNOWN`
Edge endNodeId references an existing node

`LIF_ISOLATED_NODE`
Every node is connected by at least one edge

`LIF_STATION_NODE_UNKNOWN`
Station interactionNodeIds reference existing nodes

`LIF_ROTATION_CONFLICT_AT_NODE`
Rotation allowances agree where edges meet

### LIF Vehicle Type Checks (2)

`LIF_DUPLICATE_VEHICLE_TYPE_NODE_PROPERTY`
Only one vehicleTypeNodeProperty per vehicle type per node

`LIF_DUPLICATE_VEHICLE_TYPE_EDGE_PROPERTY`
Only one vehicleTypeEdgeProperty per vehicle type per edge

### LIF Trajectory Checks (1)

`LIF_TRAJECTORY_KNOT_VECTOR_SIZE`
Trajectory knotVector has the expected size

LogIQ
VDA5050 Analyzer

# The next step

1 **Demonstration**

A walk-through on the demo package that ships with the product — one clean and one faulty data set with a matching layout. Ready to show, with no access to your installation.

2 **Pilot on your data**

One recording from your installation, analysed and discussed together. The result is a report you keep.

3 **Operation**

Hosted at vda5050.logistixpert.de or in your own network with Docker.

LogistiXpert

**Matthias Merz**
LogistiXpert
m.merz@logistixpert.de
www.logistixpert.de
vda5050.logistixpert.de