DISCLOSURE

Deviations from the official VDA5050 schemas

As of 14 August 2026 · For users, test houses and the VDA5050 working group

---

The LogIQ VDA5050 Analyzer validates captured traffic against the published JSON schemas in `github.com/VDA5050/VDA5050`. In ten places we cannot use those files as published. This document names every one of them, with its source, the passage it rests on and the reason.

We do not carry these deviations because we think the standard could be written better. We carry them because a file is otherwise unreadable, or because it contradicts the normative prose of the same specification.

The rule we follow

The specification settles the conflict itself:

> "If there are differences between the JSON schemas and this document, the variant in this document applies."
>
> — VDA5050 2.1.0, section 6 · VDA5050 3.0.0, section 7

Where schema and prose disagree, the prose wins. That is what we do. Every substantive deviation below follows the document against its own schema. None follows our opinion against both.

What was checked

Checked against the **head of the release branch**, not the tag: the branches carry the corrections made after a release is cut. `release/2.0.0` alone sits 22 commits above its tag.

Version	Branch	Commit	Commit date
1.1	`release/1.1.0` (never tagged)	`5457824`	2023-06-02
2.0	`release/2.0.0`	`33d7dc2`	2024-09-20
2.1	`release/2.1.0`	`64788b6`	2024-08-19
3.0	`release/3.0.0`	`baf900a`	2026-03-19

Fetched 2026-08-14. Every statement in this document was re-verified against these commits on that day.

Class A: the file is not valid JSON

Four published schemas carry a comma before a closing bracket. A strict parser rejects them; they cannot be read at all without intervention.

File	Version	Line	Defect
`state.schema`	2.0	220	comma before end of array
`factsheet.schema`	3.0	173	comma before end of object
`order.schema`	3.0	314	comma before end of object
`visualization.schema`	3.0	54	comma before end of object

What we do: remove the surplus comma and read the file otherwise unchanged. Nothing about the content moves. A comma before a closing bracket carries no meaning.

Class B: the schema contradicts the document

Four places where the schema demands something other than the prose of the same version. We follow the prose.

`blockingTypes` cannot be satisfied (factsheet 2.0 and 2.1)

The schema declares:

```
"blockingTypes": {
  "type": "array",
  "description": "Array of possible blocking types for defined action.",
  "enum": ["NONE", "SOFT", "HARD"]
}
```

The value has to be an array **and** equal to one of three strings. No value can be both. As published, no factsheet passes this rule, not even a perfectly correct one.

What we do: move the `enum` under `items`, which is what the field's own description calls for. VDA5050 3.0 writes it that way itself and adds the value `SINGLE` there.

`mobileRobotKinematic` is never defined (factsheet 3.0)

`typeSpecification.required` demands `mobileRobotKinematic`. The schema defines no such property. What it defines is `mobileRobotKinematics`, with an s and that is the name section 7.10 uses. The spelling without the s appears nowhere in the specification document.

What we do: point `required` at the name both the schema and the prose actually carry.

`pauseAllowed` and `cancelAllowed` have the wrong type (factsheet 3.0)

Both fields are declared `string` in the schema. The field table in section 7.10 gives both as `boolean` and their own descriptions begin `"true":` and `"false":`.

What we do: type both as `boolean`.

`requestStatus` has no value for a queued request (state 3.0)

Section 6.9 is explicit:

> "The field `requestStatus` describes the life cycle of the request and shall support the following values: […] 'QUEUED': Acknowledge the mobile robot's request to the fleet control, but no permission is given yet."

The schema lists four values for `zoneRequest.requestStatus` and `edgeRequest.requestStatus`: `REQUESTED`, `GRANTED`, `REVOKED`, `EXPIRED`. A robot reporting a queued request therefore has no legal status to report it with. The same section lets fleet control answer with exactly that.

Worth stating plainly: the field table in the same document also lists only the four. The contradiction sits inside the document. We follow the normative sentence with "shall support", not the table.

What we do: add `QUEUED` to both enums.

`PAUSED` is missing from the action statuses (state 2.1.0)

The state message table lists six values and points at the section that describes them:

> Enum {'WAITING', 'INITIALIZING', 'RUNNING', 'PAUSED', 'FINISHED', 'FAILED'} See Section 6.11 actionStates.

Section 6.11 describes `'PAUSED'` as "the action is paused because of a pause instantAction or external trigger (pause button on the AGV)". The schema's enum carries five, and its own `description` of that same field still names PAUSED — the value was taken out of the enumeration and left in the sentence beside it.

2.0.0 has all six. `startPause` is how a fleet controller holds a vehicle at a junction, and a paused vehicle otherwise has no status to report the pause with: `RUNNING` is untrue and `WAITING` means waiting for a trigger.

What we do: add `PAUSED` to the enum.

Class C: a version with no schemas at all

VDA5050 1.1 predates the `json_schemas/` folder and publishes no schemas. The folder does not exist in the branch.

What we do: derive the 1.1 schemas from 2.1.0. Mandatory fields, types and enumerations come from there, without the factsheet topic, which 1.1 does not have. These files are derived and not official and we say as much on them. Where the 1.1 field tables differ from 2.1.0, the 1.1 table governs.

What we deliberately do not do

We do not quietly improve the standard. Every deviation is written into the `$comment` of the bundled file, carries the number of its finding and lives in the tool as a rule, not as a hand-edit to a schema file.

We do not carry a correction longer than it is needed. Before every run the generator checks whether the defect is still present in the original. If it has been fixed, the tool reports the correction as expired instead of applying it. Two findings have already closed that way: the factsheet's file name in 2.0 and the field structure of that same file, both corrected in the branch head since.

We claim no certification. The analyzer produces diagnostic guidance. It issues neither certification nor release approval.

Check it yourself

The originals and the comparison ship with the product:

```
python3 tools/fetch_spec.py --all     # pull the originals from the official repo
python -m tools.sync_schemas          # report what deviates and what has expired
python -m tools.spec_field_tables --diff   # hold the field tables against the schemas
```

`VDA5050_Specs_Original/SOURCES.json` records commit, date and the files a strict parser rejects, per version. `app/validator/schemas/SOURCES.json` records commit, checksum and the corrections applied, per bundled file.

Tell us if we are wrong

If you find an error here, in our reading or in the passage we cite, tell us. We will correct the document and, where it matters, the product.

support@logiq-vda5050.com

VDA5050, M2X and LIF are the marks of their respective bodies. The LogIQ VDA5050 Analyzer is an independent tool and is neither authorised nor endorsed by them.
