OFFENLEGUNG

Abweichungen von den offiziellen VDA5050-Schemata

Stand 14. August 2026 · Für Anwender, Prüfhäuser und das VDA5050-Gremium

LogIQ Fleets prüft aufgezeichneten Verkehr gegen die veröffentlichten JSON-Schemata aus `github.com/VDA5050/VDA5050` . An dreiundzwanzig Stellen können wir diese Dateien nicht so verwenden, wie sie dort liegen. Dieses Dokument nennt jede einzelne, mit Quelle, Belegstelle und Begründung.

Wir führen diese Abweichungen nicht, weil wir den Standard für verbesserungswürdig halten. Wir führen sie, weil eine Datei sonst nicht lesbar wäre oder weil sie dem Dokument derselben Spezifikation widerspricht. An allen Stellen bis auf eine ist das der normative Fließtext. Die eine Ausnahme folgt einer Abbildung und ist als solche gekennzeichnet.

Die Regel, der wir folgen

Die Spezifikation regelt den Konfliktfall selbst:

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

Widersprechen sich Schema und Fließtext, gilt der Fließtext. Genau das tun wir. Jede inhaltliche Abweichung unten folgt dem Dokument gegen sein eigenes Schema. Achtzehn folgen einem Satz oder einer Tabelle. Eine folgt einer Abbildung: der Anfragestatus `REJECTED` . Ihr Abschnitt sagt das. Keine folgt unserer Auffassung gegen beide.

Prüfstand

Geprüft wurde gegen den **Kopf des Release-Branches**, nicht gegen das Tag: die Branches tragen die Korrekturen, die nach einem Release entstehen. `release/2.0.0` allein liegt 24 Commits über seinem Tag.

Version	Branch	Commit	Commit-Datum
1.1	`release/1.1.0` (nie getaggt)	`5457824`	02.06.2023
2.0	`release/2.0.0`	`9c45639`	28.09.2026
2.1	`release/2.1.0`	`907c183`	27.10.2026
3.0	`release/3.0.0`	`baf900a`	19.03.2026

Abgerufen am 14.10.2026, der Zweig 2.1 erneut am 10.11.2026. Sämtliche Aussagen dieses Dokuments wurden gegen diese Commits nachgeprüft. Seit der letzten Prüfung haben sich upstream zwei Zweige bewegt. In 2.0.0 nennt die Überschrift der Spezifikation die Fassung nicht mehr „Release Candidate“. In 2.1.0 lässt das Order-Schema kein negatives `allowedDeviationTheta` mehr zu; die mitgelieferten 2.1.0-Schemata folgen diesem Commit. Alle Abweichungen unten bestehen weiter.

Klasse A: die Datei ist kein gültiges JSON

Vier veröffentlichte Schemata enthalten ein Komma vor der schließenden Klammer. Ein strikter Parser lehnt sie ab; sie sind ohne Eingriff nicht einlesbar.

Datei	Version	Zeile	Fehler
`state.schema`	2.0	220	Komma vor Array-Ende
`factsheet.schema`	3.0	173	Komma vor Objektende
`order.schema`	3.0	314	Komma vor Objektende
`visualization.schema`	3.0	54	Komma vor Objektende

Was wir tun: Wir entfernen das überzählige Komma und lesen die Datei sonst unverändert ein. Der Inhalt ändert sich dadurch nicht. Ein Komma vor der schließenden Klammer trägt keine Bedeutung.

Klasse B: das Schema widerspricht dem Dokument

Neunzehn Stellen, an denen das Schema etwas anderes verlangt als das Dokument derselben Version. An achtzehn folgen wir dem Fließtext. An einer, dem Anfragestatus `REJECTED`, folgen wir einer Abbildung.

`blockingTypes` ist nicht erfüllbar (Factsheet 2.0 und 2.1)

Das Schema definiert:

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

Der Wert muss ein Array sein **und** gleichzeitig einer der drei Zeichenketten entsprechen. Beides zusammen kann kein Wert erfüllen. In dieser Fassung besteht kein einziges Factsheet die Prüfung, auch kein vollständig korrektes.

Was wir tun: Wir verschieben das `enum` unter `items`, wie es die eigene Beschreibung des Feldes verlangt. VDA5050 3.0 schreibt es selbst so und ergänzt dort zusätzlich den Wert `SINGLE`.

`mobileRobotKinematic` ist nirgends definiert (Factsheet 3.0)

`typeSpecification.required` verlangt `mobileRobotKinematic`. Dieses Feld definiert das Schema an keiner Stelle. Definiert ist `mobileRobotKinematics` mit s; so heißt das Feld auch in Abschnitt 7.10. Die Schreibweise ohne s kommt im gesamten Spezifikationsdokument kein einziges Mal vor.

Was wir tun: Wir setzen `required` auf den Namen, den Schema und Fließtext tatsächlich führen.

`pauseAllowed` und `cancelAllowed` haben den falschen Typ (Factsheet 3.0)

Beide Felder sind im Schema als `string` deklariert. Die Feldtabelle in Abschnitt 7.10 führt für beide den Typ `boolean` und ihre Beschreibungen beginnen mit `"true":` und `"false":`.

Was wir tun: Wir typisieren beide als `boolean`.

`requestStatus` kennt `QUEUED` nicht (State 3.0)

Abschnitt 6.9 ist eindeutig:

> „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."

Das Schema führt für `zoneRequest.requestStatus` und `edgeRequest.requestStatus` nur vier Werte: `REQUESTED`, `GRANTED`, `REVOKED`, `EXPIRED`. Ein Fahrzeug, das eine eingereihte Anfrage meldet, hat damit keinen zulässigen Status dafür. Dabei darf die Leitsteuerung laut demselben Abschnitt genau mit `QUEUED` antworten.

Anzumerken ist: Auch die Feldtabelle desselben Dokuments führt nur die vier Werte. Der Widerspruch liegt damit innerhalb des Dokuments. Wir folgen dem normativen Satz mit „shall support", nicht der Tabelle.

Was wir tun: Wir ergänzen `QUEUED` in beiden Enums.

Die Zonenfaktoren tragen ein Leerzeichen am Namensende (zoneSet 3.0)

Die Zweige `PRIORITY` und `PENALTY` definieren ihren Faktor als `priorityFactor` und `penaltyFactor`, mit einem Leerzeichen am Ende des Namens, während `required` in derselben Datei beide ohne Leerzeichen nennt.

Abgelehnt wird dadurch nichts, geprüft aber auch nichts. Die Norm begrenzt den Faktor auf 0,0 bis 1,0. Diese Grenzen liegen auf einer Eigenschaft, die keine Nachricht trägt; ein `priorityFactor` von 7,5 kommt unbemerkt durch. Ein Schema, das alles durchlässt, ist genau der Fehler, den ein Konformitätswerkzeug nicht erben darf.

Was wir tun: Wir entfernen das Leerzeichen aus beiden Namen. Erst dadurch gilt der von der Norm genannte Bereich für das Feld, das ein Fahrzeug tatsächlich sendet.

`PAUSED` fehlt unter den Aktionsstatus (state 2.1.0)

Die State-Tabelle führt sechs Werte und verweist auf den Abschnitt, der sie beschreibt:

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

Abschnitt 6.11 beschreibt `'PAUSED'` als „the action is paused because of a pause instantAction or external trigger (pause button on the AGV)". Das Enum des Schemas trägt fünf. Die `description` desselben Feldes nennt PAUSED weiterhin. Der Wert wurde aus der Aufzählung genommen und im Satz daneben stehen gelassen.

2.0.0 hat alle sechs. Mit `startPause` hält eine Leitsteuerung ein Fahrzeug an einer Kreuzung an. Ein pausiertes Fahrzeug hat sonst keinen Status, mit dem es die Pause melden könnte: `RUNNING` ist unwahr, `WAITING` heißt „wartet auf einen Auslöser".

Was wir tun: Wir nehmen `PAUSED` ins Enum auf.

Die Limitation heißt im Schema anders als im Dokument (zoneSet 3.0)

Tabelle 7 in Abschnitt 6.4.1.2 und die Zonentabelle in Abschnitt 7.6 nennen das Feld `directedLimitation` für eine DIRECTED-Zone und `bidirectedLimitation` für eine BIDIRECTED-Zone. Das Schema kennt in beiden Zweigen nur ein Feld `limitation` und verlangt es:

> "required": ["direction", "limitation"]

Eine nach dem Dokument geschriebene Zone wird wegen eines fehlenden `limitation` abgewiesen. Eine nach dem Schema geschriebene Zone trägt einen Namen, den das Dokument nicht kennt. Abschnitt 7 entscheidet, was gilt: „If there are differences between the JSON schemas and this document, the variant in this document applies."

Was wir tun: Wir nehmen das Feld auch unter dem Namen des Dokuments an, mit der Aufzählung seines Zweigs. Der Name des Schemas bleibt zulässig, weil eine Implementierung dem einen oder dem anderen gefolgt sein kann. Eine Zone, die keines der beiden Felder trägt, wird wie bisher gemeldet.

Die Lokalisierungs- und Navigationsarten sind geschlossene Listen (Factsheet 2.1.0)

Die typeSpecification-Tabelle in Abschnitt 6.15.1 führt beide Listen als Beispiele ein:

> localizationTypes: Simplified description of localization type. Example values: NATURAL, REFLECTOR, RFID, DMC, SPOT, GRID. navigationTypes: Array of path planning types supported by the AGV, sorted by priority. Example values: PHYSICAL_LINE_GUIDED, VIRTUAL_LINE_GUIDED, AUTONOMOUS.

Das Schema macht aus den Beispielen eine geschlossene Aufzählung auf den Einträgen beider Arrays. Ein Factsheet, das eine andere Lokalisierungs- oder Navigationsart nennt, wird abgewiesen, obwohl das Dokument keinen bestimmten Wert verlangt. Das Schema der 2.0.0 hat hier einfache Zeichenketten. 3.0 nennt beide eine erweiterbare Aufzählung.

Was wir tun: Wir entfernen die Aufzählung aus beiden Arrays. Die Einträge bleiben Zeichenketten wie in 2.0.0.

`requestStatus` kennt keinen Wert für eine abgelehnte Anfrage (State 3.0)

Abbildung 22 in Abschnitt 6.9, der Lebenszyklus einer Anfrage, zeichnet `REJECTED` als Status. Eine Anfrage erreicht ihn aus `REQUESTED` und aus `QUEUED` und endet von dort. Die Werteliste desselben Abschnitts führt ihn nicht. Die Tabellen in Abschnitt 7.8 und das Schema führen ihn auch nicht.

Das ist die eine Stelle in diesem Dokument, an der wir einer Abbildung folgen und keinem normativen Satz. Der Satz in 6.9 nennt fünf Werte, die das Feld „shall support“. `REJECTED` ist nicht darunter. Ein Fahrzeug, das nach Abbildung 22 gebaut ist, meldet aber den Status, den die Abbildung zeigt. Das Schema ließ dafür den ganzen Lauf durchfallen. Die Ursache ist ein Widerspruch im Dokument und kein Fehler des Fahrzeugs.

Was wir tun: Wir ergänzen `REJECTED` in beiden Enums. Das Schema lässt einen solchen Lauf damit nicht mehr durchfallen. Verdeckt wird der Widerspruch dadurch nicht. Unsere Regelprüfung für den Anfragestatus meldet den Wert weiter, als Minor, weil die normative Liste ihn nicht führt. Jeder andere Wert außerhalb der Liste fällt durch wie bisher.

`batteryCharging` trägt ein Leerzeichen am Namensende (Factsheet 3.0)

Das Schema definiert die Eigenschaft von `mobileRobotConfiguration` als `batteryCharging `, mit einem Leerzeichen am Ende des Namens. Abschnitt 7.10 schreibt `batteryCharging`.

Abgelehnt wird dadurch nichts, geprüft aber auch nichts. Die Typen der vier Werte und die Grenzen 0 bis 100 der drei Ladestände liegen auf einer Eigenschaft, die keine Nachricht trägt. Ein `criticalLowChargingLevel` von 750 kommt unbemerkt durch. Es ist der Fehler der Zonenfaktoren oben, in einer anderen Datei.

Was wir tun: Wir entfernen das Leerzeichen. Erst dadurch wird das Objekt geprüft, das ein Fahrzeug tatsächlich sendet.

Die Drehgeschwindigkeit einer Kante heißt im Schema anders als im Dokument (Order 3.0)

Die Kantentabelle in Abschnitt 7.3 nennt das Feld `maximumRotationSpeed`. Das Schema definiert weiter `maxRotationSpeed`, den Namen der 2.1.0. Gleich daneben hat die 3.0 `maxSpeed` im Dokument und im Schema in `maximumSpeed` umbenannt. Die Drehgeschwindigkeit wurde nur im Dokument umbenannt.

Abgelehnt wird nichts, weil das Feld optional ist. Ein Wert unter dem Namen des Dokuments wird aber gar nicht geprüft. Ein Wert unter dem Namen des Schemas trägt einen Namen, den das Dokument nicht kennt.

Was wir tun: Wir nehmen das Feld auch unter dem Namen des Dokuments an, mit demselben Typ. Der Name des Schemas bleibt zulässig, weil eine Implementierung dem einen oder dem anderen gefolgt sein kann.

`heightMin` ist im Schema nicht Pflicht (Factsheet 2.1.0)

Die physicalParameters-Tabelle in Abschnitt 6.15.1 druckt `heightMin` ohne die Kursivschrift eines optionalen Felds, wie `heightMax` darunter. Die `required`-Liste des Schemas nennt `heightMin` nicht. Ein Factsheet ohne das Feld passierte das Schema.

Was wir tun: Wir nehmen `heightMin` in `required` auf, seit dem

1. November 2026 (Fund 35). Ein 2.1.0-Factsheet ohne `heightMin` meldet die

Schemaprüfung des Factsheets als Major, auch wenn es dem veröffentlichten Schema genügt.

Wertebereiche, die das Dokument nennt und das Schema nicht (3.0)

Sieben Stellen der 3.0, an denen die Tabellen einen Bereich, einen Typ oder ein Enum nennen, das die Schemata nicht tragen. Ein Wert außerhalb des Dokuments passierte das Schema, an der ersten Stelle fiel ein richtiger Wert durch. Seit dem 19. November 2026 tragen unsere Kopien den Bereich des Dokuments:

Fund	Feld	Schema	Dokument	Wir tragen
45	`allowedDeviationXY.theta` (order)	[−π/2, π/2]	kein Bereich, Tabelle in 7.3	[−π, π] wie jeder andere Winkel
46	`headerId` (order, state, instantActions, connection, visualization, responses, zoneSet)	`integer` ohne Untergrenze	uint32, Abschnitt 7.2	Untergrenze 0, wie im Factsheet
47	`controlPoints[].weight` (order, visualization)	ohne Untergrenze	]0.0 … float64.max], Tabellen in 7.3 und 7.8	ausschließende Untergrenze 0
48	`edge.orientationType` (order)	freie Zeichenkette	Enum {'GLOBAL', 'TANGENTIAL'}, Tabelle in 7.3	dieses Enum
49	`powerSupply.batteryHealth` , `.range` (state)	`number`	int8 und uint32, Tabelle in 7.8	`integer` , Grenzen wie veröffentlicht
50	`responses[].leaseExpiry` (responses)	Zeichenkette ohne Format	Timestamp (ISO 8601, UTC), Tabelle in 7.5	`format: date-time`
51	`releaseLossBehavior` einer RELEASE-Zone (zoneSet)	nicht beschrieben	Enum {'STOP', 'CONTINUE', 'EVACUATE'}, Tabelle 6 in 6.4.1.1	dieses Enum, das Feld bleibt optional

Was wir tun: Wir setzen jeden dieser Bereiche in die gebündelte Kopie. Damit `format: date-time` wirkt, prüfen die 3.0-Validatoren seit dem

1. November 2026 Formate; die 2.x-Validatoren tun es nicht. Ein unlesbares

`leaseExpiry` meldet die Schemaprüfung der Responses als Major. Die Form des Zeitstempels im Kopf beurteilen weiter die eigenen Zeitstempel-Prüfungen und nicht das Schema. Der Entwicklungszweig für 3.0.1 schreibt jeden dieser Bereiche ebenfalls in seine Schemata.

`agvPosition.theta` hat keine Grenzen (state und visualization 2.0 und 2.1)

Die Tabelle agvPosition in 6.10.6 nennt `theta` mit „Range: [-Pi ... Pi]“. Die Schemata der 2.0 und 2.1 typisieren es als `number` ohne Grenzen, während sie `nodePosition.theta` im Order begrenzen (Fund 52).

Was wir tun: Seit dem 19. November 2026 tragen unsere Kopien die Grenzen [-Pi ... Pi]. Wo das Dokument Einheit und Bereich festlegt, ist ein Wert außerhalb ein Major.

Klasse C: eine Version ohne Schemata

VDA5050 1.1 ist älter als der Ordner `json_schemas/` und veröffentlicht keine Schemata. Der Ordner existiert im Branch nicht.

Was wir tun: Wir leiten die 1.1-Schemata aus 2.1.0 ab. Pflichtfelder, Typen und Enums kommen von dort, ohne das Factsheet-Topic, das 1.1 nicht kennt. Diese Dateien sind ausdrücklich abgeleitet und nicht offiziell. Wo die Feldtabellen von 1.1 von 2.1.0 abweichen, gilt die Tabelle aus 1.1.

Seit dem 19. November 2026 tragen die 1.1-Kopien auch die Bereiche, die die 1.1-Tabellen festlegen: `agvPosition.theta` und `nodePosition.theta` [-Pi ... Pi], `allowedDeviationTheta` [0 ... Pi], die `orientation` eines Kontrollpunkts [-Pi ... Pi] und das `weight` einer Ladung ab 0 kg.

Das gilt auch für die Schreibweise von vier Aufzählungen. Die 1.1 druckt `actionStatus`, `errorLevel` und `eStop` in ihrer Zustandstabelle (6.10.6) klein oder gemischt. Die Prozedur in 6.14 lässt das Fahrzeug `connectionState` als „online", „offline" und „connectionBroken" senden, während die Feldtabelle desselben Abschnitts Großbuchstaben führt. Die 1.1-Schemata nehmen die Schreibweise des Dokuments neben der in Großbuchstaben an und keine weitere (Fund 17). Jede dieser Stellen steht im `$comment` der jeweiligen Datei. `infoLevel` gehört nicht dazu: Die 1.1 druckt es als `{DEBUG,INFO}` und in keiner anderen Schreibweise. Angenommen werden deshalb nur Großbuchstaben.

Was wir ausdrücklich nicht tun

Wir verbessern den Standard nicht still. Jede Abweichung steht als `$comment` in der jeweiligen gebündelten Datei, trägt die Nummer des zugehörigen Fundes und ist im Werkzeug als Regel hinterlegt, nicht als Handänderung an einer Schemadatei.

Wir tragen keine Korrektur länger als nötig. Der Generator prüft vor jedem Lauf, ob der Defekt im Original überhaupt noch vorliegt. Ist er behoben, meldet er die Korrektur als überflüssig, statt sie weiter anzuwenden. Zwei Funde haben sich auf diesem Weg bereits erledigt: die Dateibenennung des Factsheets in 2.0 und die Feldstruktur derselben Datei, beide inzwischen im Branch-Kopf korrigiert.

Wir behaupten keine Zertifizierung. Der Analyzer liefert diagnostische Hinweise. Er erteilt weder eine Zertifizierung noch eine Freigabe.

Selbst nachprüfen

Die Originale und der Abgleich liegen im Produkt und lassen sich nachvollziehen:

```
python3 tools/fetch_spec.py --all     # Originale aus dem offiziellen Repo holen
python -m tools.sync_schemas          # melden, was abweicht und was überflüssig ist
python -m tools.spec_field_tables --diff   # Feldtabellen gegen die Schemata halten
```

`VDA5050_Specs_Original/SOURCES.json` hält je Version Commit, Datum und die Liste der Dateien fest, die ein strikter Parser ablehnt. `app/validator/schemas/SOURCES.json` hält je gebündelter Datei Commit, Prüfsumme und die angewandten Korrekturen fest.

Rückmeldung

Wer hier einen Fehler findet, in unserer Auslegung oder in der Belegstelle, soll ihn uns nennen. Wir korrigieren das Dokument und, wenn nötig, das Produkt.

support@logiq-fleets.com

VDA5050, M2X und LIF sind Marken der jeweiligen Gremien. Der LogIQ VDA5050 Analyzer ist ein unabhängiges Werkzeug und von diesen weder autorisiert noch anerkannt.