WolfISMS REST API

Guide for connecting external systems

Version 4.00 · as of 01 November 2026

Guide for connecting external systems to WolfISMS. The API covers six areas — **Incidents** (security incidents), **Measures**, **Assets**, **Risks**, **Suppliers** and **Vulnerabilities** — each of them for creating records, reading them, and acknowledging the synchronisation state.

Typical uses: a SIEM reports incidents into WolfISMS automatically, a ticketing system pulls open measures, a CMDB reconciles the asset inventory, a purchasing system registers new suppliers, a vulnerability scanner delivers its findings, or a reporting tool reads risks and inventory for the management report.

Contents

1. Prerequisites
2. Authentication
3. Basics
4. Incidents
5. Measures
6. Assets
7. Risks
8. Suppliers
9. Vulnerabilities
10. Synchronisation
11. Testing with curl
12. Operational notes

1. Prerequisites

Creating an API key

Every call requires an API key. An administrator creates it inside the application:

Administration → API Keys (`/admin/api_keys`)

1. Enter a description (for example "SIEM production") and create the key.
2. The key is shown **exactly once** — 64 hexadecimal characters. Copy it straight away and store it securely in the calling system.
3. WolfISMS stores only the SHA-256 hash in the database. A lost key cannot be recovered, only replaced.

Existing keys can be deactivated (`aktiv=0`) or deleted in the same screen. A deactivated key is rejected immediately — the quickest way to shut down one integration.

Base URL

```
https://<your-host>/api/v1/
```

Run this over HTTPS only. The API key travels in clear text in the header and would be readable over HTTP.

2. Authentication

The key belongs in the `X-API-Key` header:

```
curl -H "X-API-Key: 8f3c…ab12" "https://isms.example.com/api/v1/incidents"
```

If the header is missing, or the key is unknown or deactivated, the API replies with **401**:

```
{"error": "Ungültiger oder fehlender API-Key"}
```

Error messages are returned in German, matching the rest of the stored data.

No CSRF token is required — the REST API is exempt from it and relies on the key alone. There is no username-and-password login; the API has no session.

Rate limit

The REST API is **not throttled**. Sync clients may read and acknowledge every second. If you expose the API to the internet, apply a limit in the reverse proxy in front of it (for example nginx `limit_req` scoped to `/api/`).

3. Basics

Topic	Definition
Format	JSON, UTF-8
Content-Type	`application/json` (tolerated if absent when the body is read)
Dates and times	date `YYYY-MM-DD`, timestamp `YYYY-MM-DD HH:MM:SS`
Language of values	status, priority and category are stored **in German**, even when the interface is set to English
Umlauts	deliberately avoided in field names (`geaendert`, `prioritaet`)

Multiple owners

Since v3.89 a record can have several owners. `besitzer` remains the **main owner** (one user ID), `weitere_besitzer` is the list of the other owners – user IDs as well, without the main owner. This applies to all six areas:

Place	Behaviour
List and single record	return `weitere_besitzer` (an empty list if there are none); with `?besitzer_name=1` or in the single-record call the names come as `weitere_besitzer_namen`
Filter `?besitzer=`	matches records where the ID is the main **or** an additional owner
POST	`weitere_besitzer` optional as a list (`[4, 9]`), a single number or a comma-separated string; unknown IDs give 400
PATCH	`weitere_besitzer` replaces the list completely, `[]` or `null` clears it – for measures, assets, risks, suppliers and vulnerabilities, not for incidents

Permissions, tasks and reminder emails treat all owners alike; the interface shows them separated by semicolons. Names are returned for measures, assets, risks and suppliers; incidents and vulnerabilities carry the IDs only.

German values in an English interface

This is the point most integrations get wrong. Switching the interface to English translates only what is displayed — the stored value, and therefore the value the API expects, stays German. Send `in Behandlung`, not `In Progress`.

Incident status:

Send this via the API	Shown in the English interface
`Gemeldet`	Reported
`in Bewertung`	Under Assessment
`Bestätigt`	Confirmed
`Verworfen/Fehlalarm`	False Alarm
`in Behandlung`	In Progress
`Überwacht`	Monitored
`Abgeschlossen`	Completed
`Nachbearbeitung`	Follow-up
`Geschlossen`	Closed
`ohne`	– none –

Measure status:

Send this via the API	Shown in the English interface
`offen`	Open
`in Planung`	In Planning
`genehmigt`	Approved
`in Umsetzung`	In Progress
`umgesetzt`	Done
`archiviert`	Archived

Priority — `niedrig` / `mittel` / `hoch` appear as Low / Medium / High in both areas. Incidents additionally use `kritisch` (Critical), measures use `sehr hoch` (Very High). Incident categories are not translated; they appear in the interface exactly as they are stored.

Note that `in Behandlung` (incidents) and `in Umsetzung` (measures) both display as "In Progress" — they are different values in different areas.

Status codes

Code	Meaning
200	Success (GET, PATCH/PUT)
201	Record created (POST)
400	Input error — `error` names every field that was rejected
401	API key missing, unknown or deactivated
403	The module is switched off (Administration → Modules)
404	Number not found
409	Number already exists
500	Database error (details go to the server log, not into the response)

Every error response has the same shape:

```
{"error": "'prio' ist ein Pflichtfeld (auch 'prioritaet' möglich)"}
```

Multiple errors are joined with `;` , so a single call reports every complaint at once.

Field lengths

Text fields are checked against the database column width before they are written. A value that is too long comes back as **400**, naming the limit and the length supplied; it no longer surfaces as a database error (500).

Area	Field	max. characters
Incidents	`nummer` · `titel`	64 · 256
Measures	`nummer` · `name` · `abteilung` · `prioritaet`	64 · 128 · 128 · 20
Assets	`nummer` · `name` · `version` · `abteilung` · `datenhaltung`	16 · 64 · 64 · 64 · 100
Assets	`betroffene_personen` · `art_pbd` (assembled list)	500 each
Risks	`nummer` · `name` · `abteilung`	64 · 256 · 64
Suppliers	`nummer` · `name` · `ansprechpartner` · `email`	64 · 64 · 256 · 64
Vulnerabilities	`nummer` · `titel` · `quelle` · `status`	20 · 200 · 100 · 50

The free-text fields `beschreibung`, `businessimpact`, `massnahme` and `ticketnummer` hold 65,535 **bytes** – not characters: in UTF-8 an umlaut takes two bytes, an emoji four. The `beschreibung` of a measure is far larger at 16 MB.

What the body has to be

The body is a JSON **object**. An array, a bare string or a number is rejected with 400, as is an object or a list inside a text field (`{"titel": {"a": 1}}`). Numeric fields reject values the column cannot store – `kosten` for example accepts at most 9,999,999,999.99 and neither `NaN` nor `Infinity`.

4. Incidents

Method	Path	Purpose
POST	`/api/v1/incidents`	create an incident
GET	`/api/v1/incidents`	list with filters and pagination
GET	`/api/v1/incidents/<number>`	a single incident
PATCH / PUT	`/api/v1/incidents/<number>`	write status and sync state

Fields when creating

Field	Required	Values
`nummer`	yes	unique, for example `INC-00042`
`titel`	yes	short description
`prio`	yes	`niedrig` · `mittel` · `hoch` · `kritisch`
`beschreibung`	no	free text
`status`	no	default `Gemeldet`
`kategorie`	no	default `ohne`
`besitzer`	no	user ID of the main owner, validated; default `0` (none)
`weitere_besitzer`	no	list of user IDs, see section 3
`sync` / `synctime`	no	see section 10

If neither `sync` nor `synctime` is given, the API stores the incident as **synchronised**: `sync = 1`, `synctime` = time of the call. Creating a record through the interface *is* the exchange with the connected system, so no extra acknowledgement is needed. Pass `"sync": 0` to have the record picked up anyway, or `"sync": null` to exclude it entirely.

`prio` is also accepted spelled `prioritaet`, `priorität` or `prioritat`.

Status: `ohne` · `Gemeldet` · `in Bewertung` · `Bestätigt` · `Verworfen/Fehlalarm` · `in Behandlung` · `Überwacht` · `Abgeschlossen` · `Nachbearbeitung` · `Geschlossen`

Category: `ohne` and the categories the administrator maintains at `/admin/kategorien` (area Incidents). Shipped are `Malware` · `Phishing` · `Datenabfluss` · `DoS` · `Fehlkonfiguration` · `Datenschutzverletzung` · `Sonstiges`; since v3.92 the API checks against the current catalogue, so a newly created category is valid at once. The German name is stored and compared.

Status and category are case-sensitive; `prio` is lower-cased automatically.

Example: creating an incident

```
curl -X POST "https://isms.example.com/api/v1/incidents" -H "X-API-Key: $KEY" -H
"Content-Type: application/json" -d '{"nummer":"INC-00042","titel":"Phishing wave in
accounting","prio":"hoch","status":"Gemeldet","kategorie":"Phishing","beschreibung":"14
mails with a forged supplier account.","sync":0}'
```

Response **201**:

```
{
  "id": 512,
  "nummer": "INC-00042",
  "sync": 0,
  "synctime": null,
  "message": "Incident erfolgreich angelegt"
}
```

Defaults the API applies: every date field (`erkannt`, `gemeldet`, `beginn`, `beendet`, `geschlossen`, `erstellt`, `geändert`) is set to today, the person fields to 0 (no user), `typ`/`ursache`/`quelle` to `ohne`, and the GDPR/NIS2/CRA flags together with `archiviert` to 0. The interface maintains these afterwards.

Example: reading the list

```
curl -H "X-API-Key: $KEY" "https://isms.example.com/api/v1/incidents?
status=Gemeldet,in%20Behandlung&prio=hoch,kritisch&limit=20&sort=erstellt&order=desc"

{
  "total": 37,
  "count": 20,
  "limit": 20,
  "offset": 0,
  "items": [
    {
      "id": 512,
      "nummer": "INC-00042",
      "titel": "Phishing wave in accounting",
      "prio": "hoch",
      "status": "Gemeldet",
      "kategorie": "Phishing",
      "typ": "ohne",
      "ursache": "ohne",
      "quelle": "ohne",
```

```
      "erkannt": "2026-08-14",
      "gemeldet": "2026-08-14",
      "beginn": "2026-08-14",
      "beendet": "2026-08-14",
      "geschlossen": "2026-08-14",
      "erstellt": "2026-08-14",
      "geaendert": "2026-08-14",
      "archiviert": false,
      "dsgvo": false,
      "nis2": false,
      "cra": false,
      "ticketnummer": "",
      "sync": 0,
      "synctime": null,
      "besitzer": 17,
      "weitere_besitzer": [23]
    }
  ]
}
```

The description is left out on purpose — it can grow very long. Add `?beschreibung=1` to include it.

A single incident

```
curl -H "X-API-Key: $KEY" "https://isms.example.com/api/v1/incidents/INC-00042"
```

The single-record call returns the core record: `id`, `nummer`, `titel`, `beschreibung`, `prio`, `status`, `kategorie`, `erstellt`, `sync`, `synctime`, `besitzer` and `weitere_besitzer` — so the description is always included, but the extra fields of the list are not (date fields, GDPR/NIS2/CRA, ticket number). If you need the full set for one incident, query it through the list instead: `?q=INC-00042&beschreibung=1`.

An unknown number is answered with **404**.

Updating an incident

`PATCH` (or `PUT`) writes the **status** and the sync state of an existing incident. All three fields can be sent on their own or together; at least one must be present in the body:

Body	Effect
`{"status": "in Behandlung"}`	set the status; `sync` becomes 1 and `synctime` is set to now
`{"status": "Geschlossen", "sync": 0}`	set the status and keep the sync state under your own control
`{"sync": 1}`	acknowledge only, time = now
`{"sync": 0}`	reset; `synctime` stays in place
`{"sync": null}`	exclude from synchronisation
`{"synctime": null}`	clear the time only

```
curl -X PATCH "https://isms.example.com/api/v1/incidents/INC-00042" -H "X-API-Key:
$KEY" -H "Content-Type: application/json" -d '{"status": "in Behandlung"}'
```

Response **200**:

```
{
  "nummer": "INC-00042",
  "status": "in Behandlung",
  "sync": 1,
  "synctime": "2026-08-21 08:15:37",
  "message": "Vorfall aktualisiert"
}
```

`status` must be one of the values listed above, otherwise the API replies 400. Together with the status the API maintains `geändert` (today's date) and records `0` as the modifying user, which stands for the interface. Any other field in the body — `titel`, for example — produces a 400 naming the unknown keys.

The sync state is written along: if the body carries neither `sync` nor `synctime`, the API stores `sync = 1` and the current time. If the call manages either field itself, the decision stays with the client — so `{"synctime": null}` still clears the time and nothing else.

List parameters

Parameter	Effect
`status`, `prio`, `kategorie`	repeatable (`?status=A&status=B`) or comma-separated (`?status=A,B`)
`besitzer`	user ID, repeatable; matches main and additional owners
`q`	free-text search across number and title
`sync`	`0`, `1` or `null`, combinable (`?sync=0,null`)
`archiviert`	`0` (default) or `1`
`von` / `bis`	period as `YYYY-MM-DD`
`datumsfeld`	which field `von`/`bis` apply to: `erstellt` (default), `gemeldet`, `erkannt`, `beginn`, `beendet`, `geschlossen`, `geaendert`, `synctime`
`beschreibung`	`1` = include the description
`sort`	`nummer` (default), `id`, `titel`, `prio`, `status`, `kategorie`, `erstellt`, `gemeldet`, `erkannt`, `geschlossen`, `sync`, `synctime`
`order`	`asc` (default) or `desc`
`limit` / `offset`	`limit` 1–200 (default 50), `offset` from 0

5. Measures

Method	Path	Purpose
POST	`/api/v1/massnahmen`	create a measure
GET	`/api/v1/massnahmen`	list with filters and pagination
GET	`/api/v1/massnahmen/<number>`	a single measure
PATCH / PUT	`/api/v1/massnahmen/<number>`	write the ticket number and/or the sync state

The path stays `massnahmen` in both interface languages — it is part of the API, not of the display.

Fields when creating

Field	Required	Values
`nummer`	yes	unique, for example `MAS-0042`
`name`	yes	short title (alias: `titel`)
`beschreibung`	no	HTML, sanitised on the server (alias: `text`)
`status`	no	default `offen`
`prioritaet`	no	`niedrig` · `mittel` · `hoch` · `sehr hoch` (alias: `prio`)
`enddatum`	no	`YYYY-MM-DD`, defaults to today
`besitzer`	no	user ID, validated against the user table
`weitere_besitzer`	no	list of user IDs, see section 3
`abteilung`	no	free text
`abteilung_id`	no	ID from the department administration
`erledigt`	no	progress 0–100 (%)
`interner_aufwand`	no	person-days, comma or dot accepted
`kosten`	no	EUR, comma or dot accepted
`ticketnummer`	no	free text; the only business field that PATCH can change as well
`archiviert`	no	`0` (default) or 1
`sync` / `synctime`	no	see section 10

Every read response also carries the derived field `ticketsync` (`true`/`false`). It shows the yes/no field **Synchronise with ticket system** of the measure form and is `true` exactly when `sync` is not `null`. It is not written directly – use `sync` for that (see section 10).

Status: `offen` · `in Planung` · `genehmigt` · `in Umsetzung` · `umgesetzt` · `archiviert`

Two behaviours match the interface: status `umgesetzt` or `archiviert` sets `erledigt` to 100, and the review date is set to one year ahead when the record is created.

The difference between the status `archiviert` and the field `archiviert` is intentional: the first describes the state of the measure, the second removes the record from the normal view. They are independent of each other.

Read-only fields: `id`, `ersteller`, `erstellt`, `aenderer`, `geaendert`, `reviewer`, `review`, `ticketsync`, plus `besitzer_name`, `weitere_besitzer_namen` and `ersteller_name` in the format "Surname, First name".

Example: creating a measure

```
curl -X POST "https://isms.example.com/api/v1/massnahmen" -H "X-API-Key: $KEY" -H
"Content-Type: application/json" -d '{"nummer":"MAS-0042","name":"MFA for remote
access","status":"in Planung","prioritaet":"hoch","enddatum":"2026-12-
31","besitzer":17,"abteilung":"IT","erledigt":10,"interner_aufwand":"12,5","kosten":"18
000,00","ticketnummer":"CR-765898","sync":0}'
```

Response **201**:

```
{
  "id": 128,
  "nummer": "MAS-0042",
  "ticketsync": true,
  "sync": 0,
  "synctime": null,
  "message": "Maßnahme erfolgreich angelegt"
}
```

Example: reading the list

```
curl -H "X-API-Key: $KEY" "https://isms.example.com/api/v1/massnahmen?
status=in%20Umsetzung&erledigt_max=80&sort=enddatum&order=asc&besitzer_name=1"
```

One entry of the list:

```
{
  "id": 128,
  "nummer": "MAS-0042",
  "name": "MFA for remote access",
  "besitzer": 17,
  "besitzer_name": "Müller, Anna",
  "weitere_besitzer": [23],
  "weitere_besitzer_namen": ["Schmidt, Jonas"],
  "abteilung": "IT",
  "abteilung_id": 3,
  "status": "in Umsetzung",
  "prioritaet": "hoch",
  "enddatum": "2026-12-31",
  "erledigt": 80,
  "interner_aufwand": 12.5,
  "kosten": 18000.0,
  "ticketnummer": "CR-765898",
  "archiviert": false,
  "ersteller": 0,
  "erstellt": "2026-08-14",
  "aenderer": 0,
  "geaendert": "2026-08-14",
  "reviewer": 0,
  "review": "2027-08-14",
  "ticketsync": true,
  "sync": 0,
  "synctime": null
```

```
}
```

A single measure

```
curl -H "X-API-Key: $KEY" "https://isms.example.com/api/v1/massnahmen/MAS-0042"
```

Unlike incidents, the single-record call returns the **complete** record here: every field of the list plus `beschreibung`, `besitzer_name`, `weitere_besitzer_namen` and `ersteller_name` — no parameters needed.

Adding the ticket number later

Apart from the additional owners (section 3), the ticket number is the only business field that can be written after creation as well – meant for the case where the ticket system assigns the number only when it picks the measure up:

```
curl -X PATCH "https://isms.example.com/api/v1/massnahmen/MAS-0042" -H "X-API-Key:
$KEY" -H "Content-Type: application/json" -d '{"ticketnummer":"CR-765898","sync":1}'
```

Response **200**:

```
{
  "nummer": "MAS-0042",
  "ticketnummer": "CR-765898",
  "ticketsync": true,
  "sync": 1,
  "synctime": "2026-08-21 10:15:00",
  "weitere_besitzer": [23],
  "message": "Maßnahme aktualisiert"
}
```

`{"ticketnummer": null}` clears the field, `{"weitere_besitzer": [23, 31]}` replaces the additional owners. As both are business changes, the API carries the change date and the changer along, just as the interface does; changer `0` stands for the interface. It leaves the sync state alone: to acknowledge with the same call, pass `sync` as well (as in the example above). If the body only holds `sync`/`synctime`, the message stays `Sync-Status aktualisiert` as before.

List parameters

Parameter	Effect
`status`, `prioritaet`	repeatable or comma-separated; `prio` is an accepted alias
`abteilung`, `besitzer`	likewise; `besitzer` is the user ID and matches main and additional owners
`q`	free-text search across number, name and ticket number
`sync`	`0`, `1` or `null`, combinable
`archiviert`	`0` (default) or `1`
`erledigt_min` / `erledigt_max`	progress in percent
`von` / `bis`	period as `YYYY-MM-DD`
`datumsfeld`	`enddatum` (default), `erstellt`, `geaendert`, `review`, `synctime`
`beschreibung`	`1` = include the description
`besitzer_name`	`1` = include the names of owner, additional owners and creator
`sort`	`nummer` (default), `id`, `name`, `status`, `prioritaet`, `enddatum`, `erledigt`, `kosten`, `aufwand`, `abteilung`, `erstellt`, `geaendert`, `review`, `sync`, `synctime`
`order`	`asc` (default) or `desc`
`limit` / `offset`	`limit` 1–200 (default 50), `offset` from 0

Useful queries:

```
# Overdue measures that are not finished yet
curl -H "X-API-Key: $KEY" "https://isms.example.com/api/v1/massnahmen?bis=2026-08-
14&erledigt_max=99&status=offen,in%20Planung,genehmigt,in%20Umsetzung"

# All measures of one department by cost, descending
curl -H "X-API-Key: $KEY" "https://isms.example.com/api/v1/massnahmen?
abteilung=IT&sort=kosten&order=desc"
```

6. Assets

Method	Path	Purpose
POST	`/api/v1/assets`	create an asset
GET	`/api/v1/assets`	list with filters and pagination
GET	`/api/v1/assets/<number>`	a single asset
GET	`/api/v1/assets/werte`	allowed values of the master-data fields
PATCH / PUT	`/api/v1/assets/<number>`	write the sync state

Value lists come from the database

This is where assets differ from incidents and measures. The allowed values for `kategorie`, `status` and `klassifizierung` are not in the source code but in master-data tables maintained by the administration (`asset_kategorie`, `asset_status`, `klassifizierung`). What is valid therefore depends on the installation. The list for `kritikalitaet` (table `kritikalität`) only names the levels you can filter on — criticality is not set but calculated (see below).

A client should not hard-code these lists but query them:

```
curl -H "X-API-Key: $KEY" "https://isms.example.com/api/v1/assets/werte"

{
  "typ": ["primär", "sekundär"],
  "kategorie": [
    "A – Application", "B – Business Process",
    "Daten / Information", "H – Hardware",
    "I – Infrastructure", "O – Organisation", "S – System"
  ],
  "status": ["aktiv", "entsorgt", "in Beschaffung",
             "in Entsorgung", "inaktiv"],
  "klassifizierung": ["allgemein", "streng vertraulich",
                      "vertraulich", "öffentlich"],
  "kritikalitaet": ["hoch", "kritisch", "niedrig", "normal"],
  "cia_stufen": ["niedrig", "normal", "hoch", "kritisch"],
  "hinweis": "null bedeutet: Stammdatentabelle nicht vorhanden, das Feld wird ungeprüft
übernommen"
}
```

Those are the shipped defaults; entries added locally appear automatically. If one of the tables is missing, the field returns `null` and the API accepts any value for it.

Note that the endpoint name and the values themselves stay German — as everywhere else in this API, see section 3.

For the protection goals (`vertraulichkeit`, `integritaet`, `verfuegbarkeit`, `authentizitaet`) and the `wert` (value) the full scale applies: **niedrig · normal · hoch · kritisch** (low · normal · high ·

critical; `cia_stufen`).

Criticality is calculated

Since v3.94 the API no longer accepts `kritikalitaet`. Like the interface, it derives the value by the **maximum principle**: criticality is the highest level among `vertraulichkeit`, `integritaet`, `verfuegbarkeit`, `authentizitaet` and `wert`.

Protection goals and value	Criticality
all empty	`niedrig`
highest entry `normal`	`normal`
one field `hoch`, all others `niedrig`	`hoch`
one field `kritisch`	`kritisch`

Empty fields are ignored. A single high protection goal therefore determines the rating — a low value or low remaining goals do not lower it. The same rule applies when protection needs are passed on to child assets.

If a client still sends `kritikalitaet`, the field is skipped without an error, so existing integrations keep working. The response to a create call returns the calculated value. List, filter and sorting use the same value the interface shows. Records from older versions whose criticality had been set by hand or via the API are reconciled once on first access.

Fields when creating

Field	Required	Values
`nummer`	yes	unique, for example `AST-00042`
`name`	yes	short label
`version`	no	free text, e.g. `2024 R2`, `15.0.4355`
`beschreibung`	no	free text
`typ`	no	`primär` · `sekundär` (also `primaer`/`sekundaer`)
`kategorie`	no	from `asset_kategorie`
`status`	no	from `asset_status`
`klassifizierung`	no	from `klassifizierung`
`vertraulichkeit`	no	niedrig · normal · hoch · kritisch
`integritaet`	no	same scale
`verfuegbarkeit`	no	same scale
`authentizitaet`	no	same scale
`wert`	no	same scale
`besitzer`	no	user ID, validated
`weitere_besitzer`	no	list of user IDs, see section 3
`abteilung` / `abteilung_id`	no	free text / ID
`personenbezogen`	no	0 (default) or 1 — personal data involved
`betroffene_personen`	no	array or comma-separated
`art_pbd`	no	array or comma-separated
`datenhaltung`	no	free text, e.g. `On-Prem`, `Cloud EU`
`besondere_pbd`	no	0 (default) or 1 — special categories of personal data
`review`	no	`YYYY-MM-DD`, defaults to one year ahead
`archiviert`	no	0 (default) or 1
`sync` / `synctime`	no	see section 10

Some column names carry umlauts; the API uses ASCII names throughout: `integritaet` → `Integrität`, `verfuegbarkeit` → `Verfügbarkeit`, `authentizitaet` → `Authentizität`, `kritikalitaet` → `Kritik`, `personenbezogen` → `Person`, `geaendert` → `Änderdatum`.

Read-only: `id`, `kritikalitaet` (calculated), `ersteller`, `erstellt`, `aenderer`, `geaendert`, `datum`, `reviewer`, plus `besitzer_name`, `weitere_besitzer_namen` and `ersteller_name`.

Example: creating an asset

```
curl -X POST "https://isms.example.com/api/v1/assets" -H "X-API-Key: $KEY" -H "Content-
Type: application/json" -d '{"nummer":"AST-00042","name":"SAP production
system","version":"15.0.4355","typ":"primär","kategorie":"A –
Application","status":"aktiv","klassifizierung":"vertraulich","besitzer":17,"abteilung"
:"IT","vertraulichkeit":"hoch","integritaet":"hoch","verfuegbarkeit":"kritisch","person
enbezogen":1,"betroffene_personen":["Mitarbeitende"],"art_pbd":
["Stammdaten"],"datenhaltung":"On-Prem","sync":0}'
```

Response **201**:

```
{
  "id": 312,
  "nummer": "AST-00042",
  "kritikalitaet": "kritisch",
  "sync": 0,
  "synctime": null,
  "message": "Asset erfolgreich angelegt"
}
```

`kritikalitaet` is `kritisch` here because `verfuegbarkeit` is `kritisch` — the highest level decides.

`betroffene_personen` and `art_pbd` accept both a JSON array and a comma-separated string, and are stored comma-separated either way — the same format the interface writes. When read back, the API returns them as arrays.

Example: reading the list

```
curl -H "X-API-Key: $KEY" "https://isms.example.com/api/v1/assets?
kategorie=A%20%E2%80%93%20Application&status=aktiv&sort=nummer&besitzer_name=1"
```

One entry of the list:

```
{
  "id": 312,
  "nummer": "AST-00042",
  "name": "SAP production system",
  "version": "15.0.4355",
  "besitzer": 17,
  "besitzer_name": "Müller, Anna",
  "weitere_besitzer": [23],
  "weitere_besitzer_namen": ["Schmidt, Jonas"],
  "abteilung": "IT",
  "abteilung_id": 3,
  "typ": "primär",
  "kategorie": "A – Application",
  "status": "aktiv",
  "kritikalitaet": "kritisch",
  "personenbezogen": true,
```

```
  "klassifizierung": "vertraulich",
  "vertraulichkeit": "hoch",
  "integritaet": "hoch",
  "verfuegbarkeit": "kritisch",
  "authentizitaet": "",
  "wert": "",
  "betroffene_personen": ["Mitarbeitende"],
  "art_pbd": ["Stammdaten"],
  "datenhaltung": "On-Prem",
  "besondere_pbd": false,
  "archiviert": false,
  "ersteller": 0,
  "erstellt": "2026-08-14",
  "aenderer": 0,
  "geaendert": "2026-08-14",
  "datum": "2026-08-14",
  "reviewer": 0,
  "review": "2027-08-14",
  "sync": 0,
  "synctime": null
}
```


As with measures, the single-record call `/api/v1/assets/<number>` returns the complete record including `beschreibung` and the names.

List parameters

Parameter	Effect
`typ`, `kategorie`, `status`	repeatable or comma-separated
`klassifizierung`, `kritikalitaet`	likewise; `kritikalitaet` filters on the calculated value
`abteilung`, `besitzer`	likewise; `besitzer` is the user ID and matches main and additional owners
`cia`	minimum level: returns assets where confidentiality, integrity, availability **or** authenticity reaches it
`personenbezogen`	0 or 1
`q`	free-text search across number, name, description and version
`sync`	0, 1 or null, combinable
`archiviert`	0 (default) or 1
`von` / `bis`	period as `YYYY-MM-DD`
`datumsfeld`	`erstellt` (default), `geaendert`, `review`, `datum`, `synctime`
`beschreibung`	1 = include the description
`besitzer_name`	1 = include the names of owner, additional owners and creator
`sort`	`nummer` (default), `id`, `name`, `version`, `typ`, `kategorie`, `status`, `klassifizierung`, `kritikalitaet`, `abteilung`, `erstellt`, `geaendert`, `review`, `sync`, `synctime`
`order`	`asc` (default) or `desc`
`limit` / `offset`	`limit` 1–200 (default 50), `offset` from 0

`sort=kritikalitaet` orders by level (niedrig · normal · hoch · kritisch), not alphabetically.

Filtering on an unknown value is not treated as an error — it simply returns no matches. That keeps the list usable on installations that renamed their master data.

Useful queries:

```
# Assets rated high or critical, most critical first
curl -H "X-API-Key: $KEY" "https://isms.example.com/api/v1/assets?
kritikalitaet=hoch,kritisch&sort=kritikalitaet&order=desc"

# Assets with a high protection requirement in at least one goal
curl -H "X-API-Key: $KEY" "https://isms.example.com/api/v1/assets?cia=hoch&limit=100"

# Assets involving personal data that are held in the cloud
curl -H "X-API-Key: $KEY" "https://isms.example.com/api/v1/assets?
personenbezogen=1&q=Cloud"

# Assets whose review is due
curl -H "X-API-Key: $KEY" "https://isms.example.com/api/v1/assets?
```

```
datumsfeld=review&bis=2026-12-31&sort=review"
```

7. Risks

Method	Path	Purpose
POST	`/api/v1/risiken`	create a risk
GET	`/api/v1/risiken`	list with filters and pagination
GET	`/api/v1/risiken/<number>`	a single risk
GET	`/api/v1/risiken/werte`	values, scales and score bands
PATCH / PUT	`/api/v1/risiken/<number>`	write the sync state

The path stays `risiken` in both interface languages — it is part of the API, not of the display.

The assessment is calculated, not supplied

This is what sets risks apart: `riskscore` and `risiko` (the level) are derived values. A POST that includes them is rejected with 400 — otherwise a record could end up with a level that does not match its own assessment.

The API applies the same rule as the interface:

Case	Basis of the score
treatment `Akzeptieren` (accept)	values **before** measures
status `behandelt` (treated)	values **after** measures
otherwise	values **before** measures

The score is impact multiplied by likelihood. The matching level comes from the bands in the `risiko_score` table, which the administration maintains — as shipped:

Score	Level
0	nicht bewertet (not rated)
1–4	niedrig (low)
5–8	mittel (medium)
9–12	hoch (high)
13 and above	kritisch (critical)

Both values are returned by the POST, so the client never has to reproduce the calculation.

Assessment scales

`wahrscheinlichkeit_vm`, `auswirkung_vm`, `wahrscheinlichkeit_nm` and `auswirkung_nm` are integers from 0 to 5, where 0 means "not rated". The German label is accepted instead of the number:

Number	Likelihood (`wahrscheinlichkeit`)	Impact (`auswirkung`)
0	nicht bewertet — not rated	nicht bewertet — not rated
1	selten — rare	unbedeutend — insignificant
2	unwahrscheinlich — unlikely	gering — minor
3	möglich — possible	mäßig — moderate
4	wahrscheinlich — likely	groß — major
5	sicher — certain	katastrophal — catastrophic

Only the German label is accepted as input; the English wording above is what the interface displays. `vm` means "vor Maßnahmen" (before measures), `nm` means "nach Maßnahmen" (after measures).

Querying the values

Category and status come from master-data tables (`risiko_kategorie`, `risiko_status`) as they do for assets. The treatment options are fixed in the program. One call returns all of it:

```
curl -H "X-API-Key: $KEY" "https://isms.example.com/api/v1/risiken/werte"
```

The response carries `kategorie`, `status`, `behandlung`, both scales as a number-to-label mapping, the score bands and a short description of the calculation.

Treatment: `nicht bewertet` (not rated) · `Akzeptieren` (accept) · `Reduzieren` (reduce) · `Vermeiden` (avoid) · `Übertragen` (transfer) · `Teilen` (share)

Fields when creating

Field	Required	Values
`nummer`	yes	unique, for example `RSK-0042`
`name`	yes	short label
`beschreibung`	no	HTML, sanitised
`businessimpact`	no	HTML, sanitised
`massnahme`	no	HTML free text on the treatment
`kategorie`	no	from `risiko_kategorie`
`status`	no	from `risiko_status`
`behandlung`	no	default `nicht bewertet`
`wahrscheinlichkeit_vm`	no	0–5 or label
`auswirkung_vm`	no	0–5 or label
`wahrscheinlichkeit_nm`	no	0–5 or label
`auswirkung_nm`	no	0–5 or label
`besitzer`	no	user ID, validated
`weitere_besitzer`	no	list of user IDs, see section 3
`abteilung` / `abteilung_id`	no	free text / ID
`umgesetzt`	no	0 (default) or 1 — measures implemented
`review`	no	`YYYY-MM-DD`, defaults to one year ahead
`archiviert`	no	0 (default) or 1
`sync` / `synctime`	no	see section 10

Watch out for two similar names: `massnahme` is the free-text field describing the treatment, `massnahmen` is the list of linked measure records. Only `massnahme` can be written.

Read-only: `id`, `risiko`, `riskscore`, `ersteller`, `erstellt`, `aenderer`, `geaendert`, `reviewer`, plus `besitzer_name`, `weitere_besitzer_namen` and `ersteller_name`.

Links are read-only

The links to measures, assets, standards, threats, vulnerabilities, suppliers and documents are stored as comma-separated ID lists. The API returns them as arrays of numbers when reading:

```
{
  "massnahmen": [1, 3, 5, 7],
  "assets": [1, 3, 4, 14],
  "normen": [107, 158, 193]
}
```

It does **not** accept these fields when creating and rejects such a POST with 400. Writing them without checking the other side would produce links to deleted or non-existent records. Linking is done in the interface. Legacy data sometimes holds the literal text `None` in these columns; it is dropped on read.

Example: creating a risk

```
curl -X POST "https://isms.example.com/api/v1/risiken" -H "X-API-Key: $KEY" -H
"Content-Type: application/json" -d '{"nummer":"RSK-0042","name":"Ransomware via supply
chain","kategorie":"Technische Risiken","status":"in
Bewertung","behandlung":"Reduzieren","besitzer":17,"abteilung":"IT","wahrscheinlichkeit
_vm":4,"auswirkung_vm":5,"wahrscheinlichkeit_nm":2,"auswirkung_nm":3,"beschreibung":"<p
>Encryption through a compromised service provider.</p>","sync":0}'
```

Response **201** — score and level are calculated:

```
{
  "id": 88,
  "nummer": "RSK-0042",
  "riskscore": 20,
  "risiko": "kritisch",
  "sync": 0,
  "synctime": null,
  "message": "Risiko erfolgreich angelegt"
}
```

The status here is `in Bewertung`, so the values before measures apply: 4 × 5 = 20, and 20 is above 13, hence `kritisch`. If the status later changes to `behandelt` in the interface, the application computes 2 × 3 = 6 and the level drops to `mittel`.

Example: reading the list

```
curl -H "X-API-Key: $KEY" "https://isms.example.com/api/v1/risiken?
risiko=hoch,kritisch&sort=riskscore&order=desc&besitzer_name=1"
```

One entry of the list:

```
{
  "id": 88,
  "nummer": "RSK-0042",
  "name": "Ransomware via supply chain",
  "besitzer": 17,
  "besitzer_name": "Müller, Anna",
  "weitere_besitzer": [23],
  "weitere_besitzer_namen": ["Schmidt, Jonas"],
  "abteilung": "IT",
  "abteilung_id": 3,
  "kategorie": "Technische Risiken",
  "status": "in Bewertung",
```

```
  "behandlung": "Reduzieren",
  "umgesetzt": false,
  "wahrscheinlichkeit_vm": 4,
  "auswirkung_vm": 5,
  "wahrscheinlichkeit_nm": 2,
  "auswirkung_nm": 3,
  "risiko": "kritisch",
  "riskscore": 20,
  "massnahmen": [1, 3, 5],
  "assets": [14],
  "normen": [],
  "bedrohungen": [],
  "schwachstellen": [],
  "lieferanten": [],
  "dokumente": [],
  "archiviert": false,
  "ersteller": 0,
  "erstellt": "2026-08-14",
  "aenderer": 0,
  "geaendert": "2026-08-14",
  "reviewer": 0,
  "review": "2027-08-14",
  "sync": 0,
  "synctime": null
}
```


With `?beschreibung=1` the fields `beschreibung`, `businessimpact` and `massnahme` are added — three potentially long HTML texts, which is why they are left out by default. The single-record call always includes them.

List parameters

Parameter	Effect
`kategorie`, `status`, `behandlung`	repeatable or comma-separated
`risiko`	level: `nicht bewertet` · `niedrig` · `mittel` · `hoch` · `kritisch`
`abteilung`, `besitzer`	likewise; `besitzer` is the user ID and matches main and additional owners
`score_min` / `score_max`	range of the calculated score
`umgesetzt`	0 or 1
`q`	free-text search across number and name
`sync`	0, 1 or null, combinable
`archiviert`	0 (default) or 1
`von` / `bis`	period as `YYYY-MM-DD`
`datumsfeld`	`erstellt` (default), `geaendert`, `review`, `synctime`
`beschreibung`	1 = include the three text fields
`besitzer_name`	1 = include the names of owner, additional owners and creator
`sort`	`nummer` (default), `id`, `name`, `kategorie`, `status`, `behandlung`, `risiko`, `riskscore`, `abteilung`, `erstellt`, `geaendert`, `review`, `sync`, `synctime`
`order`	`asc` (default) or `desc`
`limit` / `offset`	`limit` 1–200 (default 50), `offset` from 0

Useful queries:

```
# Risks above the acceptance threshold, largest first
curl -H "X-API-Key: $KEY" "https://isms.example.com/api/v1/risiken?
score_min=9&sort=riskscore&order=desc"

# Accepted risks – for the management report
curl -H "X-API-Key: $KEY" "https://isms.example.com/api/v1/risiken?
behandlung=Akzeptieren"

# Risks whose review is due
curl -H "X-API-Key: $KEY" "https://isms.example.com/api/v1/risiken?
datumsfeld=review&bis=2026-12-31&sort=review"
```

8. Suppliers

Method	Path	Purpose
POST	`/api/v1/lieferanten`	create a supplier
GET	`/api/v1/lieferanten`	list with filters and pagination
GET	`/api/v1/lieferanten/<number>`	a single supplier
GET	`/api/v1/lieferanten/werte`	values, weights and score bands
PATCH / PUT	`/api/v1/lieferanten/<number>`	write the sync state, additional owners, description and cloud exit strategy

The path stays `lieferanten` in both interface languages — it is part of the API, not of the display.

Criteria as words, not numbers

The supplier assessment rests on eight criteria which the interface stores as an index into the `vendor_kurzbewertung` table — 0 = Ja, 1 = Nein, 2 = Teilweise, 3 = N/A. Those numbers cannot be guessed from outside, so the API takes the words instead and returns them the same way:

```
{"personendaten": "Ja", "zert": "Nein", "recht": "Teilweise", "sicher": "N/A"}
```

The values stay German, like everywhere else in this API (see section 3): `Ja` = yes, `Nein` = no, `Teilweise` = partly, `N/A` = not applicable. The number is still accepted if a client already knows it.

The eight criteria, with what they ask:

Field	Question
`personendaten`	personal data processed (GDPR)?
`schutz`	high protection requirement of the systems or data?
`recht`	legal or compliance requirements affected?
`zert`	certification in place (ISO 27001, SOC 2 …)?
`fragebogen`	security questionnaire completed?
`vertrag`	contract or data processing agreement in place?
`frueher`	earlier security incident at this supplier?
`sicher`	recorded, but not part of the calculation

Two points that easily surprise:

- **N/A counts as Ja.** That is how the interface calculates too.
- **An omitted criterion means Ja**, not "unknown" — the creation screen preselects the first entry, and the API follows it. For the damage criteria (`personendaten`, `schutz`, `recht`) `Ja` is

the worst case, so a POST carrying only `nummer` and `name` already produces a damage level of 4. Set the criteria explicitly if that is not what you want.

Tier and type

Storage differs from intuition here as well, and again the API converts:

- **`tier`** is stored as an index into `vendor_tier`, where tier 3 comes first as shipped. The API takes and returns the **tier number**: 1 = critical, 2 = standard, 3 = non-critical. The full label ("Tier 1 – Kritisch") is accepted too.
- **`typ`** refers to `vendor_typ` by ID. The API takes the **name** ("Cloud Anbieter") and returns it; the ID comes along as `typ_id`.

The assessment is calculated

As with risks, `risiko` and `risikoscore` are derived and a POST carrying them is rejected with 400. The API calculates as the interface does:

Quantity	Formula
Damage level	round((personendaten + schutz + tier + recht) / 4), minimum 1
Likelihood	round((zert + fragebogen + vertrag + frueher) / n), minimum 1
`risikoscore`	likelihood × damage level
`risiko`	level from the `risiko_score` table

The points per answer live in `vendor_gewichte` and can be adjusted per installation — as shipped the damage criteria give Ja = 5, Teilweise = 3, Nein = 0, while the likelihood criteria run the other way: Ja = 0, Teilweise = 2, Nein = 5. The tier contribution is fixed: tier 1 = 5, tier 2 = 3, tier 3 = 0 points.

The POST returns all four values, so it stays visible how the score came about.

Cloud exit strategy

Since v3.93 suppliers whose type is marked as cloud/SaaS at `/admin/vendor_typen` (`vendor_typ.cloud`) need an exit strategy (`vendor.cloud_exit`). The API provides three fields for it:

Field	Meaning
`cloud`	`true` if the type is a cloud/SaaS type
`cloud_exit`	the strategy as HTML, `""` when empty
`cloud_exit_fehlt`	`true` for cloud suppliers without a strategy

`cloud_exit` is written with POST and later with PATCH or PUT:

- **HTML** is sanitised as in the interface; script content is removed.
- **Plain text** is turned into paragraphs line by line, special characters are escaped.
- `null`, `""` or an empty paragraph delete the strategy.
- A PATCH sets the modification date and writes “API: Cloud-Exit-Strategie geändert” or “… gelöscht” to the activity log.
- Wrong types – a list instead of text, for example – are rejected with **400**.

```
curl -X PATCH "https://isms.example.com/api/v1/lieferanten/LIF-00042" -H "X-API-Key:
$KEY" -H "Content-Type: application/json" -d '{"cloud_exit":"Monthly data export as
CSV.\nDeletion confirmation after the contract ends.\nAlternative provider: Example
Ltd."}'
```

`GET /api/v1/lieferanten/werte` lists the types marked as cloud/SaaS under `cloud_typen`. If the API is the first to create the two columns, types with “Cloud” or “SaaS” in their name are preset just as through the interface.

Description

Free text about the supplier (`vendor.beschreibung`), a TinyMCE field in the interface. The API returns it as `beschreibung` (HTML, `""` when empty) and accepts it with POST and later with PATCH or PUT – under the same rules as `cloud_exit`: HTML is sanitised, plain text is turned into paragraphs line by line, `null` or `""` delete the description. A PATCH sets the modification date and writes “API: Beschreibung geändert” or “… gelöscht” to the activity log.

```
curl -X PATCH "https://isms.example.com/api/v1/lieferanten/LIF-00042" -H "X-API-Key:
$KEY" -H "Content-Type: application/json" -d '{"beschreibung":"Hosting of the customer
portals.\nData centres in Frankfurt and Berlin."}'
```

Fields when creating

Field	Required	Values
`nummer`	yes	unique, for example `LIF-00042`
`name`	yes	company name
`status`	no	from `vendor_status`
`typ`	no	name or ID from `vendor_typ`
`tier`	no	1, 2 or 3
`besitzer`	no	user ID, validated
`weitere_besitzer`	no	list of user IDs, see section 3
`ansprechpartner`	no	contact person, free text
`email`	no	free text
`personendaten`	no	Ja · Nein · Teilweise · N/A
`schutz`	no	same choice
`recht`	no	same choice
`zert`	no	same choice
`fragebogen`	no	same choice
`vertrag`	no	same choice
`frueher`	no	same choice
`sicher`	no	same choice
`beschreibung`	no	HTML or text, see Description
`cloud_exit`	no	HTML or text, see Cloud exit strategy
`review`	no	`YYYY-MM-DD`, defaults to one year ahead
`sync` / `synctime`	no	see section 10

Read-only: `id`, `risiko`, `risikoscore`, `typ_id`, `tier_bezeichnung`, `massnahmen`, `assets`, `ersteller`, `erstellt`, `aenderer`, `geaendert`, `reviewer`, `cloud`, `cloud_exit_fehlt`, plus `besitzer_name`, `weitere_besitzer_namen` and `ersteller_name`. `massnahmen` and `assets` come as arrays of numbers and are maintained in the interface only, as they are for risks.

The table has no archive flag, so there is no `archiviert` filter here.

Example: creating a supplier

```
curl -X POST "https://isms.example.com/api/v1/lieferanten" -H "X-API-Key: $KEY" -H
"Content-Type: application/json" -d '{"nummer":"LIF-00042","name":"Example Cloud
Ltd","status":"Aktiv","typ":"Cloud
Anbieter","tier":1,"besitzer":17,"ansprechpartner":"Meier","email":"contact@example.com
```

```
","personendaten":"Ja","schutz":"Ja","recht":"Teilweise","zert":"Nein","fragebogen":"Ne
in","vertrag":"Ja","frueher":"Nein","sicher":"N/A","sync":0}'
```

Response **201** — the assessment is calculated:

```
{
  "id": 42,
  "nummer": "LIF-00042",
  "eintrittswahrscheinlichkeit": 4,
  "schadenshoehe": 4,
  "risikoscore": 16,
  "risiko": "kritisch",
  "sync": 0,
  "synctime": null,
  "message": "Lieferant erfolgreich angelegt"
}
```

Worked through: damage from personendaten 5 + schutz 5 + tier 1 giving 5 + recht (Teilweise) 3 = 18, divided by 4 and rounded = 4. Likelihood from zert (Nein) 5 + fragebogen (Nein) 5 + vertrag (Ja) 0 + frueher (Nein) 5 = 15, divided by 4 and rounded = 4. Together 16 — above 13 and therefore `kritisch`.

Example: reading the list

```
curl -H "X-API-Key: $KEY" "https://isms.example.com/api/v1/lieferanten?
tier=1&sort=risikoscore&order=desc&besitzer_name=1"
```

One entry of the list:

```
{
  "id": 42,
  "nummer": "LIF-00042",
  "name": "Example Cloud Ltd",
  "personendaten": "Ja",
  "vertrag": "Ja",
  "recht": "Teilweise",
  "zert": "Nein",
  "sicher": "N/A",
  "status": "Aktiv",
  "typ": "Cloud Anbieter",
  "typ_id": 3,
  "tier": 1,
  "tier_bezeichnung": "Tier 1 – Kritisch",
  "besitzer": 17,
  "besitzer_name": "Müller, Anna",
  "weitere_besitzer": [23],
  "weitere_besitzer_namen": ["Schmidt, Jonas"],
  "fragebogen": "Nein",
  "ansprechpartner": "Meier",
  "email": "contact@example.com",
  "schutz": "Ja",
```

```
  "risiko": "kritisch",
  "risikoscore": 16,
  "frueher": "Nein",
  "massnahmen": [],
  "assets": [131, 125, 113],
  "ersteller": 0,
  "erstellt": "2026-08-14",
  "aenderer": 0,
  "geaendert": "2026-08-14",
  "reviewer": 0,
  "review": "2027-08-14",
  "beschreibung": "<p>Hosting of the customer portals.</p>",
  "cloud": true,
  "cloud_exit": "",
  "cloud_exit_fehlt": true,
  "sync": 0,
  "synctime": null
}
```


List parameters

Parameter	Effect
`status`	repeatable or comma-separated
`typ`	name or ID, repeatable
`tier`	tier number 1–3, repeatable
`risiko`	`niedrig` · `mittel` · `hoch` · `kritisch`
`besitzer`	user ID, repeatable; matches main and additional owners
`cloud`	1 = cloud/SaaS suppliers only, `0` = only the others
`cloud_exit`	`fehlt` = cloud suppliers without a strategy, `vorhanden` = with a strategy
`score_min` / `score_max`	range of the calculated score
`q`	free-text search across number, name, contact and e-mail
`sync`	0, 1 or null, combinable
`von` / `bis`	period as `YYYY-MM-DD`
`datumsfeld`	`erstellt` (default), `geaendert`, `review`, `synctime`
`besitzer_name`	1 = include the names of owner, additional owners and creator
`sort`	`nummer` (default), `id`, `name`, `status`, `typ`, `tier`, `risiko`, `risikoscore`, `besitzer`, `erstellt`, `geaendert`, `review`, `sync`, `synctime`
`order`	`asc` (default) or `desc`
`limit` / `offset`	`limit` 1–200 (default 50), `offset` from 0

Useful queries:

```
# Critical suppliers with a high risk score
curl -H "X-API-Key: $KEY" "https://isms.example.com/api/v1/lieferanten?
tier=1&score_min=9&sort=risikoscore&order=desc"

# To find suppliers without a completed questionnaire, fetch the list and
# check for fragebogen != "Ja" in the client
curl -H "X-API-Key: $KEY" "https://isms.example.com/api/v1/lieferanten?
status=Aktiv&limit=200"

# Cloud suppliers without a documented exit strategy
curl -H "X-API-Key: $KEY" "https://isms.example.com/api/v1/lieferanten?
cloud_exit=fehlt"
```

The filter runs in the database, so `total` and the paging are correct.

```
# Suppliers whose review is due
curl -H "X-API-Key: $KEY" "https://isms.example.com/api/v1/lieferanten?
datumsfeld=review&bis=2026-12-31&sort=review"
```

The individual criteria cannot be filtered on — they sit in the table as indices, and a filter on them would be exactly the numeric trap this API sets out to avoid. Fetch the list and evaluate in the client.

9. Vulnerabilities

Method	Path	Purpose
POST	`/api/v1/schwachstellen`	create a vulnerability
GET	`/api/v1/schwachstellen`	list with filters and pagination
GET	`/api/v1/schwachstellen/<nummer>`	a single vulnerability
GET	`/api/v1/schwachstellen/werte`	statuses, sources and score bands
PATCH / PUT	`/api/v1/schwachstellen/<nummer>`	write status, score, owner and sync state

The path stays German (`schwachstellen`), like every other path in this API. This area is cut for scanner operation: a vulnerability scanner, a penetration testing tool or an advisory feed delivers findings, the remediation happens in the interface, and the delivering system fetches the state back later.

The number may be omitted

Unlike the other areas, `nummer` is **optional** in a POST — a scanner does not know the WolfISMS number range. If the field is missing, the API assigns the next free number in the format `WEA-NNNNN`

and returns it in the response. A number that is supplied is checked for uniqueness as everywhere else; a duplicate ends with **409**.

Score and severity

`score` is the CVSS base score from 0.0 to 10.0 with one decimal, `null` meaning "not assessed". The severity is derived from it and not stored — it is part of every response as `schwere`:

schwere	Score
`kritisch` (critical)	9.0 and above
`hoch` (high)	7.0 up to below 9.0
`mittel` (medium)	4.0 up to below 7.0
`niedrig` (low)	above 0.0 up to below 4.0
`info` (informational)	exactly 0.0
`ohne` (none)	no score recorded

`GET /api/v1/schwachstellen/werte` returns the same bands together with the permitted statuses and sources. A client should query them there instead of hard-coding them.

Fields when creating

Field	Required	Values
`titel`	yes	short title, at most 200 characters
`nummer`	no	unique, e.g. `WEA-00042`; if omitted, the API assigns it
`beschreibung`	no	free text: affected component, CVE, technical detail, remediation
`score`	no	0.0–10.0, one decimal; `null` = not assessed
`quelle`	no	`Schwachstellenscan` · `Penetrationstest` · `Hersteller-Advisory` · `CERT/BSI-Warnung` · `Interne Meldung` · `Audit` · `Bug Bounty` · `Sonstige`
`status`	no	Neu (default) · `in Bewertung` · `Bestätigt` · `in Behandlung` · `Risiko akzeptiert` · `Behoben` · `Verworfen/Fehlalarm`
`besitzer`	no	user ID, validated
`weitere_besitzer`	no	list of user IDs, see section 3
`sync` / `synctime`	no	see section 10

Statuses and sources stay German, like everywhere else in this API (see section 3).

Read-only: `id`, `schwere`, `ersteller`, `erstellt`, `aenderer`, `geaendert` and the four link lists.

Three statuses count as finished — `Behoben`, `Risiko akzeptiert` and `Verworfen/Fehlalarm`; everything else counts as open and can be fetched with `?offen=1`. Which ones these are is also part of the `/werte` response.

Links are read-only

The links to assets, incidents, risks and measures are held as comma-separated ID lists in the table. The API returns them as arrays of numbers — always for a single record, in the list only with `?links=1`. It does not accept them when writing: a POST or PATCH carrying `assets`, `incidents`, `risiken` or `massnahmen` ends with 400. Linking happens in the interface, which checks the other side — the same rule as for risks (section 7).

Example: creating a vulnerability

```
curl -X POST "https://isms.example.com/api/v1/schwachstellen" -H "X-API-Key: $KEY" -H
"Content-Type: application/json" -d '{"titel":"Outdated OpenSSL on the web
server","beschreibung":"CVE-2026-1234, affected: web01,
web02","score":7.5,"quelle":"Schwachstellenscan","status":"Neu","besitzer":17}'
```

Response **201** — the API assigned the number, the severity is derived:

```
{
  "id": 42,
  "nummer": "WEA-00042",
  "titel": "Outdated OpenSSL on the web server",
  "score": 7.5,
  "schwere": "hoch",
  "status": "Neu",
  "sync": 1,
  "synctime": "2026-08-22 09:15:00",
  "message": "Schwachstelle erfolgreich angelegt"
}
```

Because the body carries neither `sync` nor `synctime`, the API stores `sync = 1` and the current time: writing through the interface is the exchange itself. To have the finding transferred again, set `sync` to 0 explicitly.

Example: reading the list

```
curl -H "X-API-Key: $KEY" "https://isms.example.com/api/v1/schwachstellen?
schwere=kritisch,hoch&offen=1&limit=50"

{
  "total": 2,
  "count": 2,
  "limit": 50,
  "offset": 0,
  "items": [
```

```
    {
      "id": 42,
      "nummer": "WEA-00042",
      "titel": "Outdated OpenSSL on the web server",
      "score": 7.5,
      "schwere": "hoch",
      "quelle": "Schwachstellenscan",
      "status": "Neu",
      "besitzer": 17,
      "weitere_besitzer": [],
      "ersteller": 0,
      "erstellt": "2026-08-22 09:15:00",
      "aenderer": null,
      "geaendert": null,
      "sync": 1,
      "synctime": "2026-08-22 09:15:00"
    }
  ]
}
```

`beschreibung` and the links stay out of the list unless they are requested with `?beschreibung=1` or `?links=1` — a scanner list would otherwise become unwieldy fast.

A single vulnerability

```
curl -H "X-API-Key: $KEY" "https://isms.example.com/api/v1/schwachstellen/WEA-00042"
```

Like the list, the response carries `weitere_besitzer`; it additionally holds `beschreibung` and the four link lists `assets`, `incidents`, `risiken` and `massnahmen`. An unknown number gives **404**.

Updating status, score and owner

Unlike assets, risks and suppliers, PATCH may write business fields here — the remediation state of a vulnerability often travels back from the delivering system:

```
curl -X PATCH "https://isms.example.com/api/v1/schwachstellen/WEA-00042" -H "X-API-Key:
$KEY" -H "Content-Type: application/json" -d '{"status":"Behoben","score":0}'
```

`status`, `score`, `besitzer`, `weitere_besitzer`, `sync` and `synctime` can be changed. A new main owner drops out of the additional owners so that nobody is listed twice. Any other field produces a 400 naming the unknown key. A change to a business field sets `geaendert` to now and `Aenderer` to 0, so the interface shows that the API wrote the record.

List parameters

Parameter	Effect
`status`	repeated or comma-separated
`quelle`	repeated or comma-separated
`schwere`	`kritisch` · `hoch` · `mittel` · `niedrig` · `info` · `ohne`, repeated
`besitzer`	user ID, repeated; matches main and additional owners
`score_min` / `score_max`	range of the CVSS score (0–10)
`offen`	1 = only unfinished vulnerabilities
`q`	free-text search in number and title
`sync`	0, 1 or null, combinable
`von` / `bis`	period as `YYYY-MM-DD`
`datumsfeld`	`erstellt` (default), `geaendert`, `synctime`
`beschreibung`	1 = include the description text
`links`	1 = include the link IDs
`sort`	`nummer` (default), `id`, `titel`, `score`, `status`, `quelle`, `besitzer`, `erstellt`, `geaendert`, `sync`, `synctime`
`order`	`asc` (default) or `desc`
`limit` / `offset`	`limit` 1–200 (default 50), `offset` from 0

Useful queries:

```
# The most severe open findings first
curl -H "X-API-Key: $KEY" "https://isms.example.com/api/v1/schwachstellen?
offen=1&sort=score&order=desc&limit=20"

# Everything a scan reported since 1 August
curl -H "X-API-Key: $KEY" "https://isms.example.com/api/v1/schwachstellen?
quelle=Schwachstellenscan&von=2026-08-01&datumsfeld=erstellt"

# Findings not acknowledged yet – the working set of the sync client
curl -H "X-API-Key: $KEY" "https://isms.example.com/api/v1/schwachstellen?
sync=0&limit=100"
```

10. Synchronisation

All six areas carry two columns with which an external system records what it has already taken over:

sync	Meaning
null	record is excluded from synchronisation
0	not transferred yet
1	transferred

synctime holds the time of the last transfer (YYYY-MM-DD HH:MM:SS) or is null.

For measures and incidents this state is visible in the interface as well. The measure form carries the yes/no field **Synchronise with ticket system**: unticked equals sync = null, ticking it sets sync = 0 – the form therefore never pushes an already transferred record (sync = 1) back. **Last synchronisation** is only displayed there; it is written through this interface alone.

If sync is set to 1 without passing synctime, the API fills in the current time itself. Resetting to 0 leaves synctime in place as the last known time.

How a sync client works

```
# 1. Fetch open records
curl -H "X-API-Key: $KEY" "https://isms.example.com/api/v1/massnahmen?sync=0&limit=50"

# 2. Acknowledge after a successful transfer
curl -X PATCH "https://isms.example.com/api/v1/massnahmen/MAS-0042" -H "X-API-Key:
$KEY" -H "Content-Type: application/json" -d '{"sync": 1}'
```

Response **200**:

```
{
  "nummer": "MAS-0042",
  "ticketsync": true,
  "sync": 1,
  "synctime": "2026-08-14 09:30:12",
  "message": "Sync-Status aktualisiert"
}
```

Valid bodies at a glance — each line is a complete call on its own:

Body	Effect
{"sync": 1}	acknowledge, time = now
{"sync": 1, "synctime": "2026-08-14 09:30:00"}	acknowledge with an explicit time
{"sync": 0}	reset; synctime stays as the last time
{"sync": null}	exclude from synchronisation
{"synctime": null}	clear the time only

For measures the body may also carry ticketnummer – for example {"ticketnummer":

`"CR-765898", "sync": 1}`, to write the number assigned by the ticket system back while acknowledging (section 5).

At least one of these fields must be present, otherwise the API replies 400.

For assets, risks and suppliers PATCH changes **only** the two sync fields and `weitere_besitzer` (see section 3); incidents add `status` (see section 4), measures `ticketnummer` and `weitere_besitzer` (see section 5) and vulnerabilities `status`, `score`, `besitzer` and `weitere_besitzer` (see section 9). Any other field in the body produces a 400 naming the unknown keys — so a typo surfaces immediately instead of being silently ignored. PUT behaves identically to PATCH.

Incidents and vulnerabilities set the sync state themselves on every write: if the body of a POST or PATCH carries neither `sync` nor `synctime`, the API stores `sync = 1` and the current time — writing through the interface is the exchange. The other four areas still create records with `sync = null` and only change the state when told to.

Example: a complete sync client in Python

```
#!/usr/bin/env python3
"""Fetches open measures from WolfISMS, hands them to the target system
and acknowledges every record that was transferred successfully."""

import requests

BASE = "https://isms.example.com/api/v1"
KEY  = "…"                          # load from a secure source, do not hard-code
HEAD = {"X-API-Key": KEY}


def open_measures(page=200):
    """All records with sync=0 - read completely first, then return.

    Important: reading finishes before the first record is acknowledged.
    Reading page by page while acknowledging in between would shrink the
    sync=0 result set underneath the running `offset`, and every second
    page would be skipped.
    """
    entries, offset = [], 0
    while True:
        answer = requests.get(
            f"{BASE}/massnahmen",
            headers=HEAD,
            params={"sync": 0, "limit": page, "offset": offset,
                    "beschreibung": 1},
            timeout=30,
        )
        answer.raise_for_status()
        data = answer.json()
        entries.extend(data["items"])
        offset += data["count"]
```

```
        if data["count"] == 0 or offset >= data["total"]:
            return entries


def acknowledge(number):
    answer = requests.patch(
        f"{BASE}/massnahmen/{number}",
        headers=HEAD,
        json={"sync": 1},
        timeout=30,
    )
    answer.raise_for_status()
    return answer.json()["synctime"]


for measure in open_measures():
    # ... hand the record to the target system here ...
    print(f"{measure['nummer']}: {measure['name']}")
    stamp = acknowledge(measure["nummer"])
    print(f"  acknowledged at {stamp}")
```

Acknowledge only once the target system has really accepted the record. If the run breaks before that, `sync=0` remains and the next pass picks the record up again. That is exactly why the example acknowledges one record at a time rather than all of them at the end.

For very large inventories the alternative is to request only the first page (`offset` stays 0) and repeat the call until `total` reaches 0 — then nothing has to be held in memory. This works only as long as every record that is read is also acknowledged, otherwise the loop spins forever over the same entries.

11. Testing with curl

The former REST console under Administration was removed with v4.00. An ordinary HTTP client such as `curl` is enough for trying things out; the examples in sections 4 to 10 can be sent unchanged. Best put the key into an environment variable so that it does not end up in the shell history:

```
read -rs KEY        # enter the API key, it is not echoed
curl -s -H "X-API-Key: $KEY" "https://isms.example.com/api/v1/lieferanten?limit=5"
```

`-i` additionally shows the status code and response headers. For write calls, a separate key on a test instance is advisable – a POST creates real records.

12. Operational notes

- **Number ranges:** the API does not assign numbers — with one exception. The calling system must set `nummer` itself and keep it unique; a duplicate is answered with 409. The interface uses the patterns `INC-#####`, `MAS-####`, `AST-#####`, `RSK-####`, `LIF-#####` and `WEA-#####`. The exception is vulnerabilities: if `nummer` is missing there, the API assigns the next free `WEA-#####` (section 9) — a scanner does not know the number range.
- **Schema:** the columns the API needs (`sync`, `synctime`, for measures also `Ticketnummer` and `abteilung_id`, for assets and risks `abteilung_id`) are created on the first call. No manual database update is required.
- **Asset master data:** category, status and classification come from tables the administration maintains. A client should query them via `/api/v1/assets/werte` rather than hard-coding them — see section 6. Criticality is not set by a client; the API calculates it from protection needs and value. The same applies to risks via `/api/v1/risiken/werte`, which also returns the score bands (section 7), and to suppliers via `/api/v1/lieferanten/werte`, which adds the weights from `vendor_gewichte` (section 8). For vulnerabilities, `/api/v1/schwachstellen/werte` returns the status and source lists and the boundaries of the severity bands (section 9).
- **Logging:** every write access appears in the activity feed of the application ("API: angelegt", "API: Sync-Status 1"). The user is recorded as 0, since no user is signed in.
- **Description texts:** for measures, `beschreibung` is stored as HTML and rendered in the interface. The API sanitises the text on the server; script content is removed.
- **Key rotation:** create a separate key with a meaningful description for each connected system. That way a single integration can be switched off without disturbing the others.
- **Deactivated modules:** if an administrator switches a module off under Administration → Modules, its API area is blocked as well. Calls then end with **403** and the response `{"success": false, "error": "… This module has been deactivated."}` — the key is fine, the module is not reachable. Data already recorded is preserved and can be read again unchanged once the module is switched back on.