Anleitung

Dieser Eintrag beschreibt den End-to-End-Flow, wie Datasets über die Public API Abgefragt werden können und dokumentiert die verfügbaren Filtermöglichkeiten inklusive genauer Syntax, Operatoren, Kombinationslogik, Paging und Zeitfilter.

Ablauf: Datasets finden und Daten abfragen

#	Endpoint	Zweck
1. Datasets auflisten	`GET /api/v1/datasets`	Überblick über alle verfügbaren Datasets (ID, Name, Beschreibung)
2. Dataset-Metadaten ansehen	`GET /api/v1/datasets/{datasetId}`	Details und Schema eines Datasets ansehen. Hier sind alle Kolonnen und Datentypen sichtbar (wichtig für Filterung)
3. Datensätze abfragen	`GET /api/v1/datasets/{datasetId}/data`	Daten abrufen, optional zeitlich einschränken (from/to), paginieren (limit/offset) und nach Spalten einschränken (filters)

Parameterübersicht für GET /api/v1/datasets/{datasetId}/data

URL-Parameter:

- **datasetId**: ID des Datasets (siehe Schritt 1)

Query-Parameter:

- **from**: Startdatum und Uhrzeit nach ISO 8601
  (z.B. 2025-12-02T18:30:00.000Z)
- **to**: Endatum und Uhrzeit nach ISO 8601
  (z.B. 2025-12-15T12:00:00.000Z)
- **offset**: Anzahl zu überspringender Einträge (Standard = 0)
- **limit**: Maximale Anzahl Einträge
  (Standard = 100, Maximum = 1000)
- **filters**: Kommagetrennte Liste von spalte=wert'-Paaren;
  pro Spalte sind mehrere Werte möglich (OR innerhalb einer Spalte, AND zwischen den Spalten)

Filtergrundlagen

- Ein 'filters'-String besteht aus durch Komma getrennten Einträgen der Form spalte=wert
- Mehrere Einträge mit derselben Spalte werden logisch per OR verknüpft
- Einträge für unterschiedliche Spalten werden logisch per AND verknüpft
- Die Interpretation vom Wert richtet sich automatisch nach dem Spaltentyp (wird aus dem Schema ermittelt)
    - Textspalten → LIKE oder exakte Gleichheit
    - Zahlenspalten → numerische Vergleiche inkl. Bereiche

Strings filtern

Form	Syntax	Wirkung	Kommentar
LIKE-Suche (Standard)	`spalte=wert`	`spalte LIKE '%wert%'`	Wenn im Wert bereits % gesetzt wird, bleibt das Muster unverändert
Exakte String-Gleichheit	`spalte===wert`	`spalte = 'wert'`	Hintergrund: Der Parser trennt am ersten =. Damit wert mit == beginnt, ist insgesamt === nötig.

Beispiele:

- `filters=bezirk=Bern` → `LIKE: bezirk LIKE '%Bern%'`
- `filters=bezirk===Sense` → `exakt: bezirk = 'Sense'`

Zahlen filtern

Form	Syntax	Wirkung
Exakt gleich	`spalte=wert`	`spalte = wert`
Grösser als	`spalte=>wert`	`spalte > wert`
Grösser gleich	`spalte=>=wert`	`spalte >= wert`
Kleiner als	`spalte=<wert`	`spalte < wert`
Kleiner gleich	`spalte=<=wert`	`spalte <= wert`
Ungleich	`spalte=!wert` oder `spalte=!=wert`	`spalte <> wert`
Bereich inkl. Grenzen	`spalte=wert1..wert2`	`spalte BETWEEN wert1 AND wert2`

Beispiele:

- `filters=pv_produktion_schaetzung_prognose_kwh=>=10` → Werte ab 10
- `filters=pv_produktion_schaetzung_prognose_kwh=10..20` → Werte im Bereicht 10 bis 20

- `filters=jahr=!2025` → Alle ausser 2025

Hinweis: Dezimaltrennzeichen ist der Punkt (z.B. 12.5)

Kombinationen (OR/AND)

- **Gleiche Spalte mehrfach → OR:**
  - `filters=bezirk=Bern,bezirk===Sense` → `(bezirk LIKE '%Bern%' OR bezirk = 'Sense')`
- **Unterschiedliche Spalten → AND:**
  - `filters=bezirk=Bern,pv_produktion_schaetzung_prognose_kwh=49..60` → `bezirk LIKE '%Bern%' AND pv_produktion_schaetzung_prognose_kwh BETWEEN 49 AND 60`
- **Kombiniertes Beispiel aus der Anfrage:**
  - `filters=bezirk=Bern,bezirk===Sense,pv_produktion_schaetzung_prognose_kwh=49..60` → `(bezirk LIKE '%Bern%' OR bezirk = 'Sense') AND pv_produktion_schaetzung_prognose_kwh BETWEEN 49 AND 60`

Verhalten bei ungültigen Eingaben

- Unbekannte/ungültige Spaltennamen werden abgewiesen (Statuscode: 500 Internal Server Error).
- Leere Filterwerte werden ignoriert.

Guide

This entry describes the end-to-end workflow for querying datasets via the Public API and documents the available filtering options, including exact syntax, operators, combination logic, paging, and time filters.

Finding Datasets and Querying Data

#	Endpoint	Purpose
1 List datasets	`GET /api/v1/datasets`	Get an overview of all available datasets (ID, name, description)
2 View dataset metadata	`GET /api/v1/datasets/{datasetId}`	View dataset details and schema. All columns and data types are visible here (important for filtering)
3 Query records	`GET /api/v1/datasets/{datasetId}/data`	Retrieve data, optionally restrict by time (from/to), paginate (limit/offset), and filter by columns (filters)

Parameter overview for GET /api/v1/datasets/{datasetId}/data

URL parameters:

- **`datasetId`**: ID of the datasets (see step 1)

Query-Parameter:

- **`from`**: Start date and time in ISO 8601 format (e.g., 2025-12-02T18:30:00.000Z)
- **`to`**: End date and time in ISO 8601 format (e.g., 2025-12-15T12:00:00.000Z)
- **`offset`**: Number of records to skip (default = 0)
- **`limit`**: Maximum number of records to return (default = 100, maximum = 1000)
- **`filters`**: Comma-separated list of column=value pairs; multiple values per column are allowed (OR within a column, AND between columns)

Filter basics

- A filters string consists of comma-separated entries in the form column=value
- Multiple entries with the same column are logically combined using OR
- Entries for different columns are logically combined using AND
- The interpretation of the value is automatically determined by the column type (derived from the schema):
    - Text columns → LIKE matching or exact equality
    - Numeric columns → numeric comparisons, including ranges

Filtering strings

Form	Syntax	Effect	Comment
LIKE search (default)	`column=value`	`column LIKE '%value%'`	If % is already included in the value, the pattern remains unchanged
Exact string match	`column===value`	`column = 'value'`	Background: the parser splits at the first =. For the value to start with ==, a total of === is required

Examples:

- `filters=bezirk=Bern` → `LIKE: bezirk LIKE '%Bern%'`
- `filters=bezirk===Sense` → `exact: bezirk = 'Sense'`

Filtering numbers

Form	Syntax	Effect
Exact match	`column=value`	`column = value`
Greater than	`column=>value`	`column > value`
Greater than or equal	`column=>=value`	`column >= value`
Less than	`column=<value`	`column < value`
Less than or equal	`column=<=value`	`column <= value`
Not equal	`column=!value` or `column=!=value`	`column <> value`
Range (inclusive)	`column=value1..value2`	`column BETWEEN value1 AND value2`

Examples:

- `filters=pv_produktion_schaetzung_prognose_kwh=>=10` → values from 10 and above
- `filters=pv_produktion_schaetzung_prognose_kwh=10..20` → values in the range 10 to 20

- `filters=jahr=!2025` → all values except 2025

Note: The decimal separator is a dot (e.g., 12.5)

Combinations (OR/AND)

- Same column repeated → OR
  - `filters=bezirk=Bern,bezirk===Sense` → `(bezirk LIKE '%Bern%' OR bezirk = 'Sense')`
- Different columns → AND
  - `filters=bezirk=Bern,pv_produktion_schaetzung_prognose_kwh=49..60` → `bezirk LIKE '%Bern%' AND pv_produktion_schaetzung_prognose_kwh BETWEEN 49 AND 60`
- Combined example from a request
  - `filters=bezirk=Bern,bezirk===Sense,pv_produktion_schaetzung_prognose_kwh=49..60` → `(bezirk LIKE '%Bern%' OR bezirk = 'Sense') AND pv_produktion_schaetzung_prognose_kwh BETWEEN 49 AND 60`

Behavior for invalid input

- Unknown/invalid column names are rejected (status code: **500 Internal Server Error**)
- Empty filter values are ignored