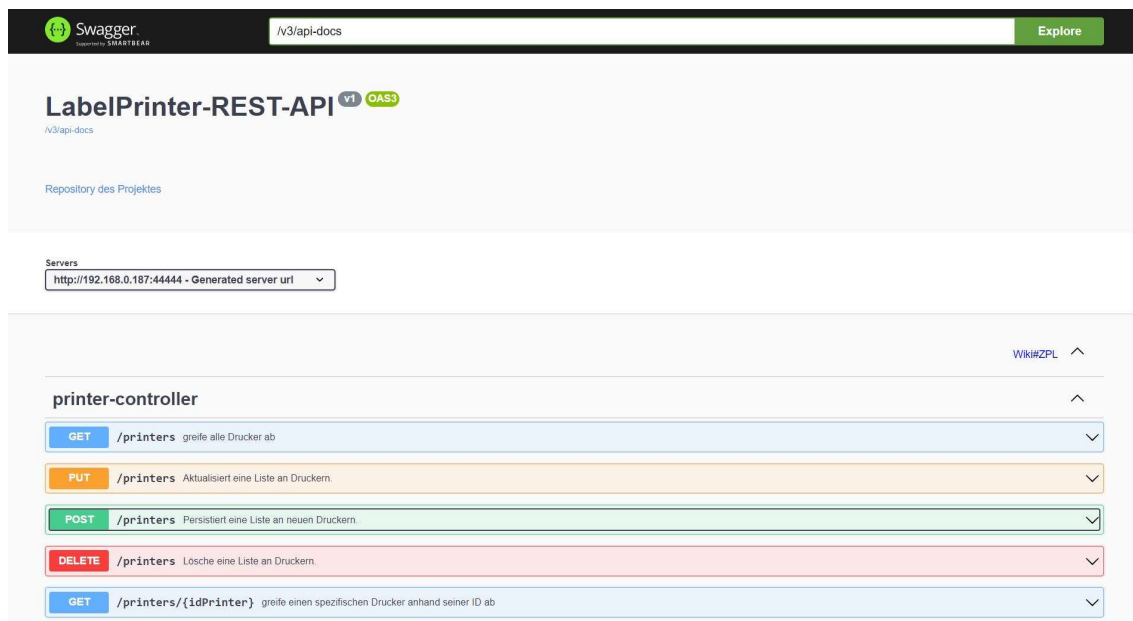


Printer- und FilewatcherBackend

- Backend-Anwendungen als Docker verfügbar.
- Einrichtung der Drucker und Druckskripte bequem über Swagger UI.
- On-Premise oder in der Cloud nutzbar.
- Ansteuerung der REST-API über JSON.
- Alternative Druckbeauftragung als zeilenbasierte Textdatei mit Freigabeordner.
- Für alle Druckerhersteller und Druckermodelle nutzbar.



Swagger
powered by SWAGGER

/v3/api-docs [Explore](#)

LabelPrinter-REST-API ^{v1} ^{OAS3}

/v3/api-docs

[Repository des Projektes](#)

Servers

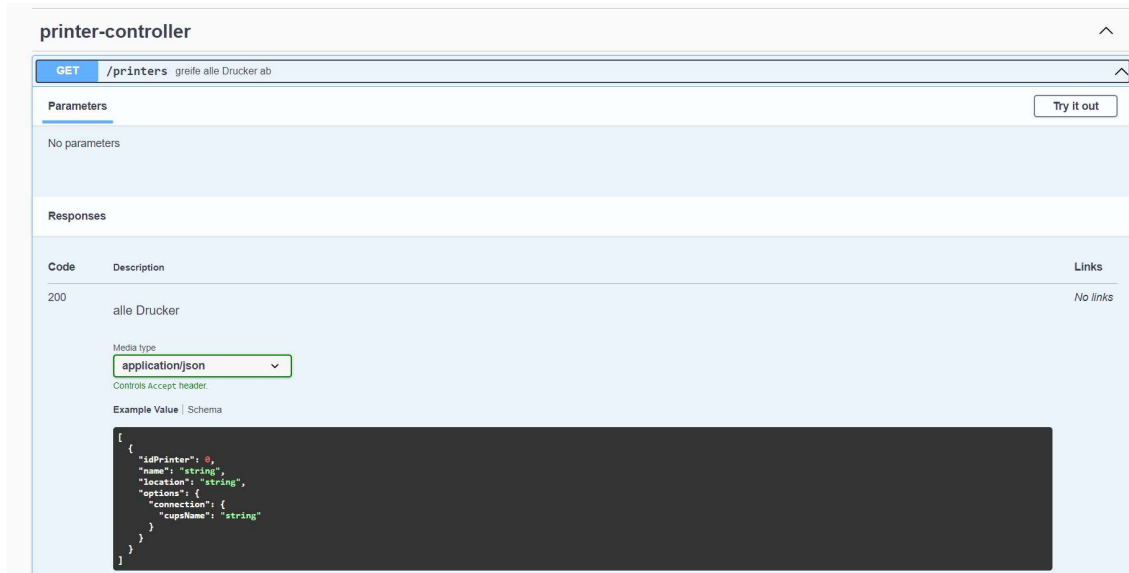
[Wiki#ZPL](#) ^

printer-controller ^

GET	/printers	greife alle Drucker ab	^
PUT	/printers	Aktualisiert eine Liste an Druckern	^
POST	/printers	Persistiert eine Liste an neuen Druckern	^
DELETE	/printers	Lösche eine Liste an Druckern	^
GET	/printers/{idPrinter}	greife einen spezifischen Drucker anhand seiner ID ab	^

Anlage Etikettendrucker

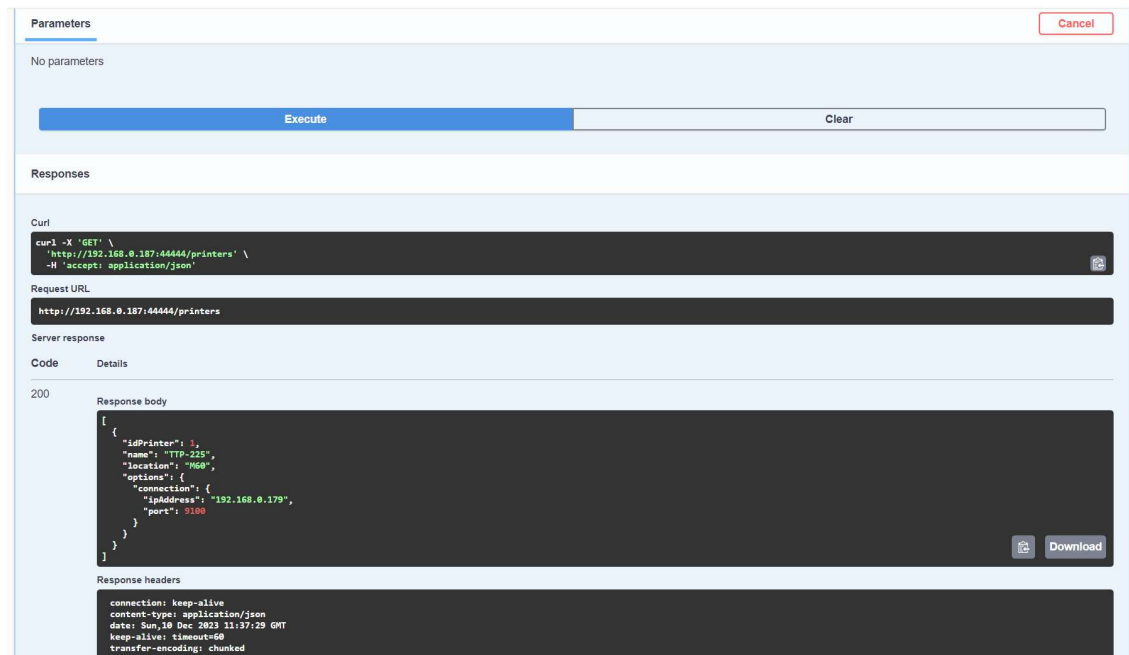
Unter der Rubrik „printer-controller“ werden die Einstellungen zu den vorhandenen Etikettendruckern durchgeführt.



The screenshot shows the 'printer-controller' REST client interface. At the top, it displays a GET request to the endpoint '/printers' with the description 'greife alle Drucker ab'. Below this, the 'Parameters' section is empty. The 'Responses' section shows a 200 status code with the description 'alle Drucker'. A dropdown menu for 'Media type' is set to 'application/json'. Below the dropdown, there is a 'Controls Accept header' section and an 'Example Value' section containing a JSON object:

```
{
  "idPrinter": 0,
  "name": "string",
  "location": "string",
  "options": {
    "connection": {
      "cupsName": "string"
    }
  }
}
```

Auflistung der installierten Etikettendrucker.



The screenshot shows the REST client interface after executing a request. The 'Parameters' section is empty. Below it, there are 'Execute' and 'Clear' buttons. The 'Responses' section shows the following details:

- Curl:**

```
curl -X 'GET' \
  'http://192.168.0.187:4444/printers' \
  -H 'accept: application/json'
```
- Request URL:** `http://192.168.0.187:4444/printers`
- Server response:**

Code	Details
200	Response body <pre>{ "idPrinter": 1, "name": "TTP-225", "location": "M68", "options": { "connection": { "ipAddress": "192.168.0.179", "port": 9100 } } }</pre>
- Response headers:**

```
connection: keep-alive
content-type: application/json
date: Sun, 18 Dec 2023 11:37:29 GMT
keep-alive: timeout=60
transfer-encoding: chunked
```

Das Anlegen von neuen Etikettendruckern im Bereich „Post“.

printer-controller ^

GET /printers greife alle Drucker ab v

PUT /printers Aktualisiert eine Liste an Druckern. v

POST /printers Persistiert eine Liste an neuen Druckern. ^

Parameters Try it out

No parameters

Request body required application/json v

alle zu speichernden Drucker

Example Value | Schema

```
[
  {
    "options": {
      "connection": {
        "cupName": "string"
      }
    },
    "name": "string",
    "location": "string"
  }
]
```

Responses

PUT /printers Aktualisiert eine Liste an Druckern. v

POST /printers Persistiert eine Liste an neuen Druckern. ^

Parameters Cancel Reset

No parameters

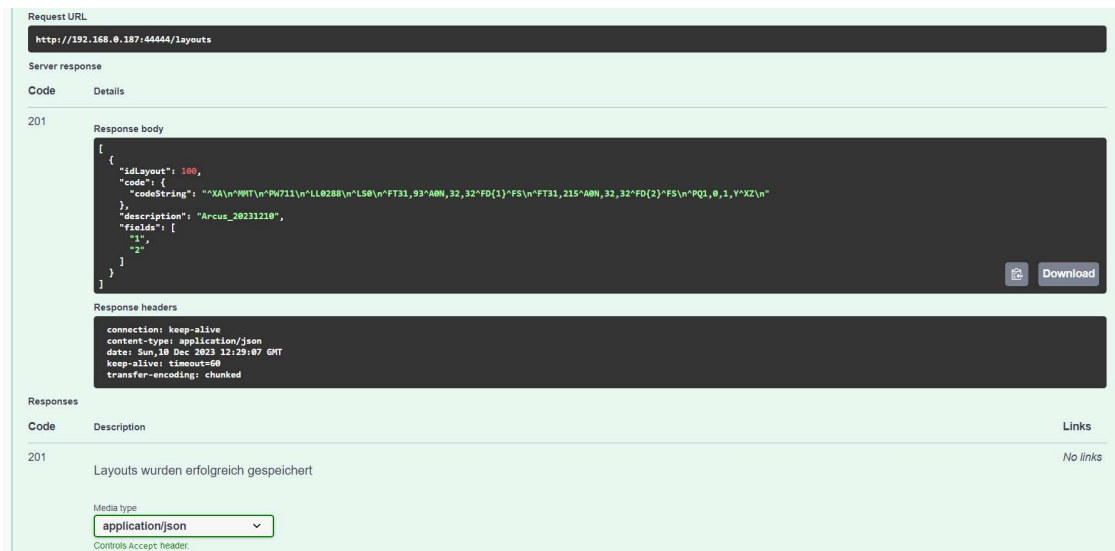
Request body required application/json v

alle zu speichernden Drucker

```
[
  {
    "options": {
      "connection": {
        "port": 9100,
        "ipAddress": "192.168.0.180",
        "cupName": "string"
      }
    },
    "name": "Z0421 1",
    "location": "Ecodistrict"
  }
]
```

Execute

Nach Fertigstellung der Anlage wird dem Layout eine ID zugewiesen.



The screenshot shows a REST client interface with the following details:

- Request URL:** http://192.168.0.187:4444/layouts
- Server response:** 201
- Response body (JSON):**

```
[{"idLayout": 100, "code": {"codeString": "XA\\nWT\\nPW711\\nL6288\\nLS0\\nFT31,93*AM, 32, 32*FD(1)*FS\\nFT31, 215*AM, 32, 32*FD(2)*FS\\nPQ1, 0, 1, Y*XZ\\n"}, "description": "Arcus_20231210", "fields": [{"id": "1", "value": "2"}]}]
```
- Response headers:**

```
connection: keep-alive
content-type: application/json
date: Sun, 10 Dec 2023 11:23:07 GMT
keep-alive: timeout=60
transfer-encoding: chunked
```
- Responses table:**

Code	Description	Links
201	Layouts wurden erfolgreich gespeichert	No links
- Media type:** application/json

„idPrinter“ und „idLayout“ in Bezug auf das FilewatcherBackend

Das FilewatcherBackend ist ein separater optionaler Service, welcher in der Lage ist Textdateien mit zeilenbasierten Inhalten zu verarbeiten.

```
version: '3'

services:
  filewatcher-app:
    image: scaleit/filewatcher-app
    container_name: filewatcher-app
    restart: unless-stopped
    volumes:
      - /home/sb/filewatcher:/data
    environment:
      - PRINTER_ID=129
      - LAYOUT_ID=100
      - WATCHED_DIRECTORY=/data
      - LABELPRINTER_URL=http://host.docker.internal:4444
      - PYTHONUNBUFFERED=1
    extra_hosts:
      - "host.docker.internal:host-gateway"

volumes:
  data:
```

Ansteuerung mittels JSON über die REST-API

Das PrinterBackend lässt sich auch direkt über eine Übergabe eines JSON-Requests an die REST-API benutzen. Hierfür ist ein entsprechend formatierter JSON-Request an die REST-API zu senden.

```
Curl
curl -X 'POST' \
  'http://192.168.0.187:4444/printers/1/print/3' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d [
    {
      "identification": "string",
      "value": "string"
    }
  ]
Request URL
http://192.168.0.187:4444/printers/1/print/3
```



Weitere Informationen unter:

YUMA Technologie GmbH

Siemensstraße 2

72184 Eutingen im Gäu

Tel.: +49 7459 93043-0

E-Mail: info@yuma-technologie.com

<https://www.yuma-technologie.com>