LEXITANON

LexITanon Server
REST API Guide

Connecting document management systems, scripts and other software to LexITanon Server

Program version	LexITanon 0.5.0 (model 1.3.3, rules 4)
API version	v1 (all addresses start with `/api/v1`)
Edition	LexITanon Server (Linux and Windows Server)
Date	November 2026
Publisher	be next GmbH, Kirchheimer Str. 27, 70619 Stuttgart, Germany · https://lexitanon.com

Contents

1 About this guide
2 Starting the server for API use
3 Conventions
4 Endpoints at a glance
5 Redacting text directly
6 Processing files as jobs
7 Options
8 Custom rules
9 Server information and administration
10 Status codes and errors
11 Placeholders and categories (glossary)

1 About this guide

LexITanon pseudonymises and redacts personal data in German legal documents (PDF and text). Everything runs on your own server: documents are never sent to be next or anyone else. LexITanon Server offers all functions of its browser interface through a REST API (JSON over HTTP), so that document management systems, mail gateways or your own scripts can use it.

This guide describes the API of LexITanon Server 0.5.0. Installation, licensing and the browser interface are described in the user manual (German), available at https://lexitanon.com/handbuch/.

> **Language.** LexITanon works on German documents. Its user interface, the texts of error messages and warnings, the placeholders (such as `[NAME]` or `[DATUM]`) and the identifiers of the categories (such as `personen`) are German. Field names, error codes and option values are fixed identifiers and do not change with a translation; your software should rely on them, not on message texts. Chapter 11 translates all categories and placeholders. The machine-readable description `GET /api/v1/openapi.json` (OpenAPI 3.1) has German descriptions in version 0.5.0 and English ones from the next version on; this guide describes the same interface.

Desktop edition. The API described here is meant for LexITanon Server. LexITanon Desktop uses the same API internally for its own window, but accepts requests only from its own page. Access from other computers and the simultaneous processing of several documents through interfaces require LexITanon Server (terms of use, clause 4.7 and 4.8).

2 Starting the server for API use

LexITanon Server listens on two ports: the API port (default **8471**) and the port of the browser interface (default **8470**), which also answers API requests. By default it accepts connections from the server itself only. All settings are start parameters (`--name value` or `--name=value`); `lexitanon --help` lists them all.

Parameter	Default	Meaning
`--bind <ip>`	`127.0.0.1`	network address; `0.0.0.0` = all network interfaces
`--api-port <port>`	`8471`	port of the REST API; `0` = API only on the interface port

`--ui-port <port>`	`8470`	port of the browser interface (also serves the API); `0` = no interface
`--workers <n>`	cores / 4, at least 1	documents processed at the same time
`--threads-per-worker <n>`	`4`	processor cores the AI model uses per document
`--queue-size <n>`	`100`	waiting jobs; short texts have the same number of places of their own. Beyond that the server answers 503
`--max-connections <n>`	4 × cores, at least 16	simultaneous connections
`--max-upload-mb <n>`	`150`	largest file in MB (larger: 413)
`--result-ttl <minutes>`	`15`	results are deleted at the latest after this time; every access to a job extends it
`--footer <text|none>`	standard text	default footer of result PDFs for requests without their own choice
`--custom-rules <file>`	next to the licence file	file of the stored custom rules (chapter 8)
`--log-level <level>`	`info`	`debug`, `info`, `warn` or `error`

Examples:

```
# API only, reachable from the internal network, no browser interface
lexitanon --bind 0.0.0.0 --ui-port 0

# a large server with many parallel requests
lexitanon --bind 0.0.0.0 --workers 8 --threads-per-worker 4 --max-connections 100

# larger files, results kept for one hour
lexitanon --max-upload-mb 300 --result-ttl 60
```

On Windows Server the installer sets up the service “LexITanon”; other parameters are set as administrator with `lexitanon.exe --install-service` followed by the parameters (the licence is kept).

> **Network security.** LexITanon does not ask for a login and transfers data unencrypted (HTTP). Make it reachable only within a trusted internal network, never from the internet. If other systems must reach it across network boundaries, put it behind a reverse proxy with TLS and authentication, or restrict access with a firewall.

3 Conventions

- **Base address:** `http://<server>:8471/api/v1`. The examples use `API=http://localhost:8471/api/v1`.
- **Encoding:** requests and responses in JSON are UTF-8. Text files must be UTF-8 (a byte order mark is allowed).
- **Positions:** `start` and `end` of a finding count Unicode characters (code points, not bytes) in the field `text` of the same response. `text` is the processed text after normalisation, which may differ slightly from the text you sent (for example in line breaks).

- **Errors** always have the form `{"error": {"code": "...", "message": "..."}}`; licence errors add `reason` (chapter 10).
- **Licence:** documents are processed only while the licence of the installation is valid; otherwise every processing request answers 402.
- **Browsers of other websites:** requests that a browser sends on behalf of another website (with a foreign `Origin` header) are refused with 403. Server-to-server calls send no such header and are not affected.
- **Local only:** licence management, the session log and saving custom rules are possible only from the server itself (`127.0.0.1` or `::1`); other computers get 403 `local_only`.

4 Endpoints at a glance

Request	Purpose
`POST /redact-text`	process a short text immediately; the result is in the response
`POST /jobs`	submit a file (PDF or UTF-8 text) as a job
`GET /jobs/{id}`	status of the job, place in the queue, findings
`GET /jobs/{id}/result`	download the result as PDF, text or JSON
`POST /jobs/{id}/result`	create a PDF or text file from a corrected result text
`DELETE /jobs/{id}`	delete the job and its result immediately
`GET /settings`	categories (values for `groups`) and replacement modes
`GET /info`	version, AI model, rules, licence status, limits, queue
`GET /custom-rules`	read the stored custom rules
`PUT /custom-rules`	save the custom rules (from the server itself only)
`POST /custom-rules/test`	try rules on a sample text
`POST /licence/activate`, `/licence/deactivate`, `/licence/check`	licence management (from the server itself only)
`GET /update`	whether a newer version is available
`GET /session-log`	session log for support (from the server itself only)
`POST /feedback`	optional: send an example of a wrong recognition to be next (section 9.5)
`GET /openapi.json`	machine-readable description (OpenAPI 3.1)

5 Redacting text directly

`POST /api/v1/redact-text` processes a text and returns the result at once. It suits short texts such as e-mails or text fields; for files and long documents use jobs (chapter 6), which are queued reliably even under heavy load.

Request (file `request.json`):

```
{"text": "Max Mustermann, geboren am 3. Mai 1980, wohnt Hauptstraße 5, 10115 Berlin."}
```

```
curl -X POST $API/redact-text -H "Content-Type: application/json" -d @request.json
```

Response (200):

```
{
  "redacted": "[NAME], geboren am [DATUM], wohnt [ANSCHRIFT], [ORT].",
  "text": "Max Mustermann, geboren am 3. Mai 1980, wohnt Hauptstraße 5, 10115 Berlin.",
  "findings": [
    {"start": 0,  "end": 14, "group": "personen", "key": "personen",  "label": "PER",
     "original": "Max Mustermann", "replacement": "[NAME]"},
    {"start": 27, "end": 38, "group": "datum",    "key": "datum",     "label": "DATUM",
     "original": "3. Mai 1980",    "replacement": "[DATUM]"},
    {"start": 46, "end": 59, "group": "orte",     "key": "anschrift", "label": "ANSCHRIFT",
     "original": "Hauptstraße 5",  "replacement": "[ANSCHRIFT]"},
    {"start": 61, "end": 73, "group": "orte",     "key": "plz",       "label": "ORT",
     "original": "10115 Berlin",   "replacement": "[ORT]"}
  ],
  "warnings": []
}
```

Field	Meaning
`redacted`	the result text
`text`	the processed (normalised) input; `start` / `end` refer to it
`findings`	one entry per replaced passage: position, category `group` (chapter 11), original and replacement. `key` and `label` are technical details for support requests
`warnings`	notes in German, for example about a date without an exact day
`date_shift_days`	only with `"dates": "shift"` : the shift used, in days (section 7.4)

The request accepts all options of chapter 7, for example:

```
{"text": "Der Kläger Max Mustermann verklagt die Beispiel GmbH. Herr Mustermann trägt vor,
          er habe am 10.05.2024 gekündigt.",
 "numbering": true, "dates": "shift", "date_shift_days": 14}

→ "redacted": "Der Kläger [NAME_1] verklagt die [FIRMA_1]. Herr [NAME_1] trägt vor,
               er habe am 24.05.2024 gekündigt."
```

6 Processing files as jobs

Files are submitted as jobs. LexITanon queues them and processes as many at the same time as `--workers` allows; this stays reliable with many simultaneous requests. A job goes through the states `queued` → `running` → `done` or `failed`.

6.1 Submit, poll, download, delete

```
API=http://localhost:8471/api/v1

# 1. submit, optionally with options (chapter 7)
curl -F "file=@Urteil.pdf" \
     -F 'options={"groups": ["personen", "organisationen", "orte", "datum", "aktenzeichen"],
                  "replacement": "label", "numbering": true}' \
     $API/jobs
# answer (202): {"job_id": "dc009322c1bba5455f6d33fa6f1c9f65", "status": "queued", "queue_position": 1}

JOB=dc009322c1bba5455f6d33fa6f1c9f65

# 2. poll until "status" is "done" (or "failed"), e.g. once per second
curl $API/jobs/$JOB

# 3. download the result and delete the job
curl -o Urteil_pseudonymisiert.pdf "$API/jobs/$JOB/result?format=pdf"
curl -X DELETE $API/jobs/$JOB
```

The multipart field `file` is required; `options` is an optional JSON text. Instead of multipart you can send the file itself as the request body (`Content-Type: application/pdf`, `text/plain` or `application/octet-stream`) with the query parameters `filename`, `replacement` and `select` (technical keys, comma-separated). This simple form supports only these three parameters; use multipart for all other options.

```
curl --data-binary @Urteil.pdf -H "Content-Type: application/pdf" \
     "$API/jobs?filename=Urteil.pdf&replacement=block"
```

6.2 Job status

`GET /jobs/{id}` returns:

Field	Meaning
`job_id`, `status`	the job and its state: `queued`, `running`, `done`, `failed`
`queue_position`	while queued: its place in the queue
`filename`, `size`, `type`	the submitted file, its size in bytes, `pdf` or `txt`
`pages`, `output_pages`	pages of the input PDF and of the result PDF
`replacement`	`label` or `block`
`created`, `finished`, `seconds`	times (UTC) and processing time
`findings`, `warnings`	as in chapter 5
`error`	with `failed`: `code`, `message` and the HTTP `status` that applies, e.g. 422 for a scanned PDF without text layer
`date_shift_days`	only with `"dates": "shift"`

6.3 Downloading the result

`GET /jobs/{id}/result` with the query parameter `format`:

- `pdf` (default): the result as a new PDF. Placeholders are highlighted in colour; `colors=0` gives a PDF without highlighting, for printing.
- `txt`: the result as UTF-8 text.
- `json`: `{"text", "redacted", "findings", "warnings"}`, with the text read from the file.

As long as the job is not finished, the answer is 409; try again later. Results are deleted automatically after `--result-ttl` minutes (default 15); every access to a job extends this time. Delete jobs yourself as soon as you have the result.

6.4 Corrected results

If a user corrects the result text (for example replaces a name LexITanon missed), `POST /jobs/{id}/result` turns the corrected text into a file with the job's file name, footer and placeholder colours. The text is not stored.

```
curl -X POST $API/jobs/$JOB/result -H "Content-Type: application/json" \
     -d '{"redacted": "Sehr geehrter Herr [NAME], ...", "format": "pdf", "colors": true}' \
     -o corrected.pdf
```

6.5 Load and retries

When the queue is full, the server answers 503 with a `Retry-After` header (seconds); repeat the request after that time. `GET /info` shows the current load under `queue` (`running`, `queued`, `stored`) and the limits under `limits`.

6.6 PowerShell and Python

Windows PowerShell 7:

```
$api = "http://localhost:8471/api/v1"
$job = Invoke-RestMethod -Method Post "$api/jobs" -Form @{ file = Get-Item .\Urteil.pdf }
do {
    Start-Sleep 1
    $s = Invoke-RestMethod "$api/jobs/$($job.job_id)"
} until ($s.status -in "done", "failed")
Invoke-WebRequest "$api/jobs/$($job.job_id)/result?format=pdf" -OutFile Urteil_pseudonymisiert.pdf
Invoke-RestMethod -Method Delete "$api/jobs/$($job.job_id)"
```

Python 3 with the `requests` package:

```
import json, time, requests

API = "http://localhost:8471/api/v1"
options = {"replacement": "label", "numbering": True, "footer": "none"}
with open("Urteil.pdf", "rb") as f:
    r = requests.post(f"{API}/jobs", files={"file": f}, data={"options": json.dumps(options)})
r.raise_for_status()
job = r.json()["job_id"]
while True:
    s = requests.get(f"{API}/jobs/{job}").json()
    if s["status"] in ("done", "failed"):
        break
    time.sleep(1)
if s["status"] == "done":
    pdf = requests.get(f"{API}/jobs/{job}/result", params={"format": "pdf"})
    open("Urteil_pseudonymisiert.pdf", "wb").write(pdf.content)
else:
    print(s["error"]["code"], s["error"]["message"])
requests.delete(f"{API}/jobs/{job}")
```

7 Options

All options are optional. They are accepted by `POST /redact-text` (in the JSON body) and by `POST /jobs` (in the multipart field `options`). Unknown options are refused with 400 `bad_options`.

7.1 What is replaced: groups and select

`groups` is a list of category identifiers from `GET /settings` (chapter 11), for example `["personen", "orte", "kontakt"]`. Without `groups` all categories marked as default are replaced. `select` is an alternative with technical keys (the `keys` list of `/settings`); use either `groups` or `select`, not both. The group `eigene` stands for the stored custom rules.

7.2 How it is replaced: replacement

Value	Result
`label` (default)	pseudonymisation: placeholders such as `[NAME]`, `[FIRMA]`
`block`	redaction: black bars (█) of the same length

7.3 Numbered placeholders: numbering

`"numbering": true` numbers the placeholders: `[NAME_1]`, `[NAME_2]`, `[FIRMA_1]`. The same value keeps its number throughout the document, and a part of a name ("Mustermann" after "Max Mustermann") gets the number of the full name. Not with `block` and not for custom rules (`[CUSTOM_n]` already carries the rule number).

7.4 Dates: dates and date_shift_days

dates	Result
`replace` (default)	dates become `[DATUM]`

`keep`	dates stay as they are
`month`	only month and year remain (“Mai 2024”)
`shift`	all dates move by the same number of days, in their original format, so intervals between dates are kept

With `shift`, `date_shift_days` sets the shift (a whole number from −3650 to 3650, not 0). Without it LexITanon picks a random shift of 4 to 52 whole weeks and reports it in `date_shift_days`; send that value with further documents of the same case to keep them consistent. Dates of birth are always replaced, whatever the setting. A date without an exact day is replaced as with `replace` and noted in `warnings`.

7.5 Footer of the result PDF: footer and footer_text

footer	Result
`default`	the standard footer, or the server's own default set with `--footer`
`none`	no footer text; page numbers remain
`custom`	the text in `footer_text` (one line, at most 150 characters), e.g. “Anonymisiert durch Kanzlei Beispiel”

The footer also applies to corrected results (section 6.4).

7.6 Custom rules for one request: custom_rules

`custom_rules` is a list of up to 20 rules (format as in chapter 8) that apply to this request only, in addition to the stored rules. Their numbers must differ from those of the stored rules. They apply even if `groups` does not contain `eigene`.

7.7 Example

```
{"text": "…", "groups": ["personen", "organisationen", "datum"], "replacement": "label",
 "numbering": true, "dates": "shift", "footer": "custom",
 "footer_text": "Anonymisiert durch Kanzlei Beispiel"}
```

8 Custom rules

Custom rules (“Eigene Regeln”) recognise your own identifiers that the AI model cannot know, such as your firm's file numbers, client numbers or project names. Each rule has a number from 1 to 99; its matches are replaced by `[CUSTOM_1]`, `[CUSTOM_2]` and so on. Exceptions name text that is never replaced, not even by the AI model, for example your firm's own address. Stored rules apply to every request to the server; they are kept in `custom-rules.json` next to the licence file (or in the file given with `--custom-rules`).

8.1 Kinds of rules

kind	Description
`pattern`	simple pattern: `#` one digit, `@` one letter (umlauts included), `*` 1 to 40 characters without spaces, a space matches any whitespace, `\` makes the next character literal. Optional `prefix`: fixed text before the match that stays. Example: `####-####-@@` finds “2024-0815-MK”.

`words`	a list of fixed words or names (up to 1000), e.g. project names
`regex`	regular expression (PCRE2) for experts; `group` replaces only this capture group (0 = whole match). `\w`, `\d` and `\b` are ASCII only; use `\p{L}` for letters with umlauts
`exception`	exactly one of `words`, `pattern` or `regex`: anything inside a match is never replaced. No `number`

8.2 Fields of a rule

Field	Meaning
`number`	1 to 99, gives the placeholder `[CUSTOM_number]`; required except for exceptions
`kind`	`pattern`, `words`, `regex` or `exception` (required)
`name`	a name for the rule (up to 80 characters)
`pattern`, `prefix`	pattern (up to 200 characters) and optional fixed text before it (up to 100)
`words`	list of words (each up to 200 characters)
`regex`, `group`	regular expression (up to 500 characters) and capture group
`ignore_case`	ignore upper and lower case (default `false`)
`active`	`false` keeps the rule stored but switched off (default `true`)
`examples`	up to 10 sample texts; each must be found by the rule, otherwise the rule is refused

8.3 Reading, saving and testing

`GET /custom-rules` returns `{"available", "editable", "format": 1, "rules": [...]}`; `editable` is true when the request comes from the server itself. `PUT /custom-rules` replaces all stored rules (from the server itself only). Every rule and its examples are checked first; if one fails, nothing changes (400 or 422 with the reason).

```
curl -X PUT http://localhost:8471/api/v1/custom-rules -H "Content-Type: application/json" -d '{
  "format": 1,
  "rules": [
    {"number": 1, "name": "Unser Aktenzeichen", "kind": "pattern",
     "prefix": "Unser Zeichen:", "pattern": "####-####-@@",
     "examples": ["Unser Zeichen: 2024-0815-MK"]},
    {"kind": "exception", "name": "Eigene Kanzlei", "words": ["Kanzlei Beispiel"]}
  ]}'
```

`POST /custom-rules/test` tries rules on a sample text without changing anything: `{"text": "...", "rules": [...]}`; without `rules` the stored rules are used. The answer lists what the rules find (`findings`) and what an exception keeps (`excepted`):

```
{"text": "Unser Zeichen: 2024-0815-MK, Kanzlei Beispiel",
 "findings": [{"start": 15, "end": 27, "label": "CUSTOM_1",
               "original": "2024-0815-MK", "replacement": "[CUSTOM_1]"}],
 "excepted": [], "warnings": []}
```

9 Server information and administration

9.1 GET /info

Returns `name`, `version`, `edition` (`server`), the AI model (`model.release`), the recognition rules (`rules.version`), `self_test`, the licence (`license`: `status` `valid` or `invalid`, `code`, `edition`, `term_end`, `valid_until` and more), the `limits` (`max_upload_mb`, `workers`, `queue_size`, `result_ttl_minutes`, `max_connections`) and the current `queue`. Use it as a health check: the server is ready when it answers 200 and `license.status` is `valid`.

9.2 GET /settings

Returns the categories (`groups`: `id`, German `name`, `placeholders`, `default`), the technical `keys` and the replacement modes. Chapter 11 translates the categories.

9.3 Licence

From the server itself only: `POST /licence/activate` with `{"licence_key": "LXK-XXXXX-XXXXX-XXXXX"}` activates the installation (the server then restarts), `POST /licence/deactivate` releases it (for example before moving to another server), `POST /licence/check` checks it at once. The Server edition checks its licence automatically every hour; the check transfers licence data only, never document content.

9.4 Updates and support log

`GET /update` reports whether a newer version is available. On the Server edition the administrator installs it on the command line (`sudo lexitanon --update` on Linux, `lexitanon.exe --update` as administrator on Windows). `GET /session-log` (from the server itself only) returns the session log for support: the start-up checks and events since then, without document content.

9.5 POST /feedback (optional)

“Fehlerhafte Erkennung melden” is a voluntary function: a user may send an example of a wrong recognition to be next to improve LexITanon. It is not needed for processing and is never called automatically. The example must contain invented data only. Body: `kind` (`missed`, `too_much` or `wrong_kind`), `text` (1 to 5000 characters), optional `expected` (category) and `versions`. Answers: 200 `{"sent": true}`; 429 after 50 reports per installation and day; 503 without a valid licence or connection.

10 Status codes and errors

Code	Meaning
200 / 202 / 204	success / job accepted / deleted
400	invalid request, e.g. an unknown category in `groups` or an unknown option (`bad_options`)
402	no valid licence; `reason` names the cause (below)
403	request from another website, a local-only function called from another computer (`local_only`), or, with `--bind 127.0.0.1`, not via localhost
404	job unknown or already deleted (`not_found`)

409	result not ready yet; ask again later
413	file larger than `--max-upload-mb`
415	file type not supported (PDF and UTF-8 text only)
422	PDF without a text layer (scanned document), or custom rules that fail their check
429	`/feedback` only: daily limit reached
503	busy, the queue is full: repeat after `Retry-After` seconds

Example of an error:

```
{"error": {"code": "bad_options",
           "message": "Unbekannte Gruppe in groups: foo (siehe GET /api/v1/settings)."}}
```

Values of `reason` with 402 and licence errors: `not_activated`, `other_machine`, `clock`, `offline`, `duplicate`, `released`, `expired`, `waitpay`, `pending`, `update_required`, `suspended`, `cancelled`, `seats`, `unknown_key`, `network`. The German `message` explains what to do; `GET /info` shows the licence details.

11 Placeholders and categories (glossary)

The identifiers in the first column are the values for `groups` and appear as `group` in every finding.

group	Category	Placeholders (meaning)
`personen`	persons	`[NAME]` name of a person, `[ANWALT]` lawyer
`organisationen`	companies and organisations	`[FIRMA]` company, law firm, bank or public authority
`orte`	addresses and places	`[ANSCHRIFT]` street address, `[ORT]` postcode and town or place
`datum`	dates	`[DATUM]` date, `[GEBURTSDATUM]` date of birth
`aktenzeichen`	file numbers of the own proceedings	`[AKTENZEICHEN]` court or case file number
`kontakt`	contact details	`[E-MAIL]` e-mail address, `[TELEFON]` phone or fax number, `[WEBADRESSE]` web address
`bank`	bank details	`[IBAN]` account number (IBAN)
`steuerid`	tax identification number	`[STEUER-ID]` German tax ID
`kennnummern`	identification numbers	`[SAFE-ID]` beA/SAFE identifier, `[UST-IDNR]` VAT ID, `[REGISTERNR]` commercial register number, `[KENNZEICHEN]` vehicle registration, `[IP-ADRESSE]` IP address, `[FIN]` vehicle identification number, `[KENNUNG]` insurance or customer number
`gesundheit`	health data	`[DIAGNOSE]` diagnosis, `[EINRICHTUNG]` hospital or practice, `[KENNUNG]` medical identifier
`eigene`	custom rules (chapter 8)	`[CUSTOM_1]`, `[CUSTOM_2]` … by rule number

What stays on purpose: courts and their chambers (“Landgericht Frankfurt am Main, 4. Zivilkammer”), amounts of money, laws and provisions (“§ 823 Abs. 1 BGB”), cited decisions with their date and file number (“BGH, Urteil vom 12.03.2020 - VIII ZR 12/19”), and roles such as Kläger (claimant), Beklagte (defendant) or Zeuge (witness).

The categories may grow with new versions of the recognition rules; `GET /settings` always returns the current list.