



Rate limits & temporary blocks

What happens when your integration asks too much, and how to build it so it never does.

TL;DR

A **rate limit** is a maximum on how many requests your integration may send in a certain amount of time. Send too many too fast and the API answers **429 Too Many Requests** instead of your data. It protects the platform so it stays fast for everyone.

Got a 429?

Wait the number of seconds in **Retry-After**, then continue slower.

Best fix

Use webhooks instead of polling, and send a **User-Agent** that says who you are.

Keep ignoring 429s?

Your key is blocked for a while.

Writing your own API client?

Use our feature-complete **MCP server** instead:
<https://mcp.reuzenpanda.app/mcp>.

01 Building something custom?

FEATURE-COMPLETE MCP SERVER

For AI assistants and agents (Claude, ChatGPT, Copilot, ...) there is no need to write your own API client: connect the Reuzenpanda MCP server. It is feature-complete and handles authentication for you.

Endpoint <https://mcp.reuzenpanda.app/mcp>

Docs <https://docs.reuzenpanda.app/ai-integration>

CUSTOM SOLUTIONS

Building a custom integration or solution on top of Reuzenpanda? Contact support before you go live. We check with you that it fits our terms of use, and if the public API does not cover what you need, **we will think along with you about a solution**.

02 Building an efficient integration

★ Register webhooks instead of polling

Subscribe to the events you need (for example *Quotation Created*, *Invoice Created*, *Contract Created*, *Status Updated*, *Lead/Deal Created*, *Deal Won/Lost*) via **Webhook Subscriptions** in the API reference on api.reuzenpanda.app, and fetch an item only when you are notified about it. Re-reading every document on each run is the most common cause of 429s.

No webhook for your case?

Contact support and tell us which event you need. We would rather add a webhook than have your integration poll for changes.

Use an API key for integrations

Not the login of a user. Scripts running on a user login are treated as an integration anyway, and a key makes it clear to both of us which integration is calling. Building a native integration that users connect to their own account? Contact support to request an **OAuth client**.

Spread scheduled jobs

Avoid starting large syncs at exactly the same moment (on the hour, first thing in the morning); a job that runs steadily over time stays within budget far more easily than one burst.

One job at a time per key

Parallel workers share the same budget and only reach the limit sooner. **Log 429 responses**, so you notice you are close to your budget before a block happens.

Need more capacity, or unsure why you receive 429s?

Contact Reuzenpanda support with the time of the requests, the endpoint and the **name of your API key or integration** (never the key itself). We are happy to look at your integration with you.

03 What happens when you send too much

2xx

All good

A budget per key

Each API key has a request budget. Short peaks are fine; the budget refills continuously, so a steady integration never runs out.

normal response

WHAT TO DO

Nothing. Where applicable, X-RateLimit-Limit and X-RateLimit-Remaining show your budget for that endpoint.

429

Slow down

Over the budget: wait

You receive **429** with **Retry-After**. Wait that many seconds, then continue at a slower pace.

```
{"message": "RATE_LIMITED",
 "retryAfterSeconds": n}
```

WHAT TO DO

Wait Retry-After seconds (always present, 1 s to ~1 h), then retry that request. Other endpoints may still work.

429

Blocked

Keep going: temporary block

Keep sending after repeated 429s and the key is **blocked on all endpoints** for a while. It lifts automatically; repeats last longer.

```
{"message": "RATE_LIMITED_BLOCKED",
 "retryAfterSeconds": n}
```

WHAT TO DO

Stop all requests with this key until Retry-After has passed. Retrying earlier does not help: every request is refused until then.

04 Handling a 429 correctly

Read Retry-After, pause, and retry the same request. Do not fire the next request in your loop straight away, and do not spread the load over parallel threads.

PYTHON

```
import time, random, requests

session = requests.Session()
session.headers["User-Agent"] = "acme-crm-sync/1.4 (contact: dev@acme.nl)"

def get(session, url, max_attempts=5):
    for attempt in range(max_attempts):
        r = session.get(url, timeout=30)
        if r.status_code != 429:
            return r
        wait = int(r.headers.get("Retry-After", "5"))
        if r.json().get("message") == "RATE_LIMITED_BLOCKED":
            raise RuntimeError(f"API key blocked, pause the whole job for {wait}s")
        time.sleep(wait + random.uniform(0, 1)) # small jitter
    raise RuntimeError("still rate limited, try again later")
```

05 Identify your integration

Send a descriptive User-Agent on every request, so we can tell integrations apart and contact the right people when something goes wrong:

HEADER

```
User-Agent: <integration-name>/<version> (contact: <email>)
User-Agent: acme-crm-sync/1.4 (contact: dev@acme.nl)
```

Requests that do not identify the developer or integration (for example a default python-requests, Python-urllib or node-fetch User-Agent) may receive a tighter budget.

06 Good to know

After a block, your key is on probation

The budget comes back only partly, and far fewer 429s are enough for the next, longer block. Do not restart at full speed the moment a block lifts: resume slowly and honour every Retry-After.

Blocks lift automatically

A block applies to the calling integration on **all endpoints**. On some endpoints the budget is shared by all keys of the same account, so splitting one job over several keys does not raise it.

Repeat offenders are flagged

Integrations that are blocked repeatedly are flagged. We will contact you, and if an integration keeps overloading the platform we may temporarily disable its key.

Limits can change

Limits are set per endpoint and may be adjusted over time. Rely on the response headers and Retry-After, not on fixed numbers.