Design: Rate Limiting for the Export API

Engineering design document · RFC-0142

Platform Team

2026-10-20

Summary

The export API has no per-client limit today. A single client running a backfill can take most of the worker pool and slow everyone else down. This document proposes a **token-bucket rate limiter** at the gateway, with limits set per API key.

> **Important**
>
> Status: **approved** for implementation in the November release.

Goals and non-goals

Goals

- Keep the export API's p95 latency below 800 ms when one client sends ten times its usual load.
- Give every client a clear, documented limit and a standard `429` response with `Retry-After`.

Non-goals

- Billing based on usage — handled separately in RFC-0137.
- Limits inside the export workers themselves.

Design

Each API key has a bucket of capacity C that refills at r tokens per second. A request costs one token; an export of n pages costs $\lceil n/50 \rceil$ tokens. The number of tokens available at time t is

$$B(t) = \min\bigl(C,\; B(t_0) + r\,(t - t_0)\bigr) \tag{1}$$

A request is accepted when $B(t)$ in Eq. (1) is at least its cost.

Table 1 Default limits per plan

Plan	Capacity C	Refill r (tokens/s)	Burst allowed
Free	20	0.5	40 s of quota
Team	200	5	40 s of quota
Enterprise	2,000	50	40 s of quota

The gateway keeps the buckets in Redis and updates them atomically:

```
-- KEYS[1] = bucket key, ARGV = capacity, rate, now (s), cost
local c, r, now, cost = tonumber(ARGV[1]), tonumber(ARGV[2]),
 ↪  tonumber(ARGV[3]), tonumber(ARGV[4])
local b = redis.call("HMGET", KEYS[1], "tokens", "ts")
local tokens = math.min(c, (tonumber(b[1]) or c) + r * (now - (tonumber(b[2])
 ↪  or now)))
if tokens < cost then return {0, math.ceil((cost - tokens) / r)} end
redis.call("HSET", KEYS[1], "tokens", tokens - cost, "ts", now)
return {1, 0}
```

Limits are configured per plan (Table 1) and can be raised for a single key:

```
rate_limits:
  overrides:
    key_9f3a: { capacity: 5000, rate: 120 }   # nightly migration, until
 ↪  2026-12-31
```

Rollout

1. Ship in **shadow mode**: compute the decision, log it, never reject.
2. After one week, compare shadow rejections with support tickets.
3. Enforce for the Free plan, then Team and Enterprise one week apart.

Alternatives considered

Option	Why not
Fixed window counter	Allows double the limit at window edges.
Limits in each worker	Uneven across workers; a client could still flood the queue.
Concurrency limit only	Does not protect against many short requests.