V9 Rate Limits

This page defines the default rate limits for the backend services. The limits are separated by context to offer granular control over limits within the application area.

What are rate limits?

Rate limiting is a mechanism used to protect an application from excessive traffic by controlling how many requests can be processed over a period of time. It helps prevent system overload, abuse, and performance degradation by limiting request rates, controlling traffic bursts, or restricting the number of concurrent operations. When a limit is reached, requests may either be placed in a queue and processed later or be rejected with an **HTTP 429 (Too Many Requests)** response. Different rate limiting strategies are available depending on the scenario: **Fixed Window** limits requests within a fixed time period, **Sliding Window** provides smoother traffic control using a moving time window, **Token Bucket** allows short bursts while enforcing a sustainable average rate, and **Concurrency** limits the number of requests that can execute at the same time. All queued requests in this implementation are processed using an **OldestFirst** approach.

Rate Limiting Configuration

Global Behaviour

- Rejected requests return **HTTP 429 (Too Many Requests)**.
- Response trailer includes:
  - `error_detail: too many requests`
- Queue processing order for all policies:
  - **OldestFirst**

Built-in Rate Limiter Policies

Policy Name	Type	Purpose	Configuration
fixed	Fixed Window	Allows a fixed number of requests within a time period. Counter resets at the end of each window.	`PermitLimit=100`, `Window=20s`, `QueueLimit=50`
sliding	Sliding Window	Similar to Fixed Window but uses a moving time window for smoother traffic control.	`PermitLimit=25`, `Window=9s`, `SegmentsPerWindow=3`, `QueueLimit=10`

Policy Name	Type	Purpose	Configuration
token	Token Bucket	Uses tokens that are consumed by requests and replenished over time. Allows short bursts of traffic.	`TokenLimit=50`, `TokensPerPeriod=1`, `ReplenishmentPeriod=5s`, `AutoReplenishment=true`, `QueueLimit=10`
concurrency	Concurrency	Limits the number of requests that can execute simultaneously.	`PermitLimit=2`, `QueueLimit=3`

Limiter Type Comparison

Limiter Type	What It Limits	Example
Fixed Window	Number of requests in a fixed time period	Allow 100 requests every 20 seconds
Sliding Window	Number of requests in a continuously moving time period	Allow 25 requests within any rolling 9-second period
Token Bucket	Requests based on available tokens that refill over time	Allow bursts of requests but enforce a sustainable rate
Concurrency	Number of requests running simultaneously	Allow only 2 imports to execute at the same time

Token Bucket Business Policies

Configuration values are sourced from application settings, with the defaults shown below.

Policy Name	Configuration Section	Purpose	Token Limit	Tokens / Period	Replenishment Period	Queue Limit
api-policy	`ApiRateLimitPolicy:*`	General API request throttling. Allows small bursts while protecting the API from excessive traffic.	60	10	10s	10
import-policy	`ImportRateLimitPolicy:*`	Supports high-volume import operations while preventing imports from overwhelming the system.	1,000	200	10s	20
signify-signing-policy	`SigningRateLimitPolicy:*`	Designed for high-throughput document signing workloads.	4,000	1,000	5s	10

Policy Name	Configuration Section	Purpose	Token Limit	Tokens / Period	Replenishment Period	Queue Limit
signify-email-policy	`EmailRateLimitPolicy:*`	Controls email sending throughput and protects downstream email providers.	6,000	1,000	10s	50
signify-sms-policy	`SMSRateLimitPolicy:*`	Limits SMS traffic to avoid overwhelming SMS gateways and third-party providers.	1,000	100	10s	10

What This Means in Practice

E.g. the **api-policy** for API requests:

- Handle a **burst of up to 60 requests immediately**.
- Recover at a rate of **10 requests every 10 seconds** (approximately 1 request per second on average).
- Queue up to **10 additional requests** while waiting for tokens to become available.
- Reject further requests with **HTTP 429** when both the bucket and queue are full.

This configuration is useful because it allows short traffic spikes while still protecting the API from sustained high request volumes. The same token bucket policy is enforced for the other policies defined above.

Queue Behaviour

Scenario	Result
Permit available	Request executes immediately
Permit unavailable, queue has space	Request waits in queue
Permit unavailable, queue full	Request rejected with HTTP 429
Multiple requests queued	Processed in `OldestFirst` order

Summary

All rate limiters:

- Return **HTTP 429** when requests are rejected.
- Include the trailer `error_detail: too many requests`.
- Process queued requests using **OldestFirst** ordering.

Business-specific policies use the **Token Bucket** algorithm and are configurable on application level.

Revision #7
Created 2026-06-24 11:12:49 UTC by Nardus van Eyk
Updated 2026-06-24 11:48:53 UTC by Nardus van Eyk