Throttling Guidance

ABSTRACT

Throttling limits the number of concurrent calls to a service to prevent overuse of resources. SalesTim API is designed to handle a high volume of requests. If an overwhelming number of requests occurs, throttling helps maintain optimal performance and reliability of the SalesTim API service.
Throttling limits vary based on the scenario. For example, if you are performing a large volume of writes, the possibility for throttling is higher than if you are only performing reads.

TABLE OF CONTENTS

[[toc]]

What happens when throttling occurs?

When a throttling threshold is exceeded, SalesTim API limits any further requests from that client for a period of time. When throttling occurs, SalesTim API returns HTTP status code 429 (Too many requests), and the requests fail. A suggested wait time is returned in the response header of the failed request.

SalesTim API is conforming to the IETF ratelimit standardization proposal.

> Note: Throttling behavior can depend on the type and number of requests. For example, if you have a high volume of requests, all requests types are throttled. Threshold limits vary based on the request type. Therefore, you could encounter a scenario where writes are throttled but reads are still permitted.

Common throttling scenarios

The most common causes of throttling of clients include: - A large number of requests across all applications in a our environments. - A large number of requests from a particular application across all environments.

Best practices to avoid throttling

Programming patterns like continuously polling a resource to check for updates and regularly scanning resource collections to check for new or deleted resources are more likely to lead to applications being throttled and degrade overall performances.

Before any throttling, SalesTim API provides two useful headers included in every responses so that you can monitor your own activity level: - `X-RateLimit-Limit`: The limit of requests in a perdiod of time (aka "window") - `X-RateLimit-Remaining`: The current number of requests that could be made during the current window.

Best practices to handle throttling

The following are best practices for handling throttling:

- Reduce the number of operations per request.
- Reduce the frequency of calls.
- Avoid immediate retries, because all requests accrue against your usage limits.

When you implement error handling, use the HTTP error code 429 to detect throttling. The failed response includes the `Retry-After` response header. Backing off requests using the `Retry-After` delay is the fastest way to recover from throttling because SalesTim API continues to log resource usage while a client is being throttled.

1. Wait the number of seconds specified in the `Retry-After` header.

2. Retry the request.
3. If the request fails again with a 429 error code, you are still being throttled. Continue to use the recommended `Retry-After` delay and retry the request until it succeeds.

   If no `Retry-After` header is provided by the response, we recommend implementing an exponential backoff retry policy.

In addition to the `Retry-After` header, SalesTim API includes `X-RateLimit-Limit` and `X-RateLimit-Remaining` infos in body of the throttled response:

```
{
  message: 'Too many requests, please try again later...',
  rateLimitExceeded: {
    tier: 'Tier 1',              # Could be 'Tier 1', 'Tier 2' or 'Tier 3'
    rateLimitWindow: 900000,     # In ms
    rateLimitMax: 6              # In # of requests
  }
}
```