API Best Practices

Kibo recommends some best practices when interfacing with the Kibo Composable Commerce Platform (KCCP) APIs to improve system efficiency and minimize potential problems such as server load and slow performance. Your tenant should be set up to follow these guidelines for the best KCCP experience.

This guide explains the request rate limiting that is enforced by the platform as well as additional best practices for interacting with the APIs.

Rate Limiting

You should self-manage your own tenant to avoid overburdening the KCCP system and negatively impacting performance for all other users. This is enforced by Kibo's rate limiting rules, which will reject requests from any tenants that are submitting too many requests to certain API routes within a given time frame.

In the sandbox environment, all tenants are grouped together under the same rate limits. If one tenant reaches the limit, then requests on all sandboxes will be rejected. However, tenants are rate limited separately in production environments. If one production tenant reaches a limit, then the restriction will only affect that specific tenant. Rate limits for staging and pre-prod environments also apply per tenant just like production tenants. Rate limits for sandbox, pre-prod, performance testing, and production do not affect each other and are counted separately.

HTTP Response

The maximum limit can vary depending on the route and is distributed across your entire tenant at any given time. When this limit is reached, subsequent requests will be rejected with an HTTP 429 "Too Many Requests" status code:

```
HTTP/1.1 429 Too Many Requests
Date: Fri, 3 May 2022 00:19:56 GMT
Content-Length: 0
Connection: keep-alive
Server: nginx/1.15.8
Retry-After: 60
x-vol-correlation: faac10ae0c224fa089bf6b1fe6305c5a
```

At this point you should wait before submitting any more requests because the system will reject them until the time has elapsed. The time you should wait is indicated in the response header:

Response Header	**Description**

Retry-After	The remaining rejection time, or the amount of seconds you should wait before placing another request. Values will be 60, 900, 1800, 2700, or 3600 which correspond to 1 minute, 15 minutes, 30 minutes, 45 minutes, and 60 minutes.

Acceptable Request Rate

You can determine an acceptable rate of requests by finding the rule that best matches the API route and dividing the rule's *limit* by the *time period*. All applications or client code that make a request to an API route with a matching rule must self-manage their request rate to stay at or below the limit to avoid receiving an HTTP 429 error.

For example, if a rule for an API has a limit of 100 requests per minute (RPM) and 2,000 requests per hour (RPH) then the maximum number of requests allowed is 100 per minute and 2000 per hour.

If your requests are being rejected by KCCP, then your application or client code must wait for the restriction time to expire (per the Retry-After header) and then lower the rate of requests to avoid hitting the limit again.

Minute vs. Hourly Limiting

Limits on a given rule are applied per minute and per hour. A single request to the KCCP system will count against both the minute and hourly limit for the matching rule. For the examples below, assume the rule is an API with limits of 100 requests per minute (RPM) and 2,000 requests per hour (RPH).

Minute limits count the number of requests in a single minute. If 120 requests are made in 60 seconds, then the system will begin restricting requests on the 101st attempt and return HTTP 429 responses for all further requests until the wait time has expired. Once the next minute begins, the per-minute limit is reset and requests will be accepted unless the rate limit is reached again.

Hourly limits operate on a rolling 60-minute window, which is broken down into four 15-minute buckets. Starting from the first request, the system counts the number of requests in 15-minute buckets. For example, if 100 requests per minute (RPM) come in for 20 minutes then the system will reject any further requests for up to 45 minutes since the entire hourly limit was used. For another example, if exactly 30 requests per minute (RPM) come in during the first 45 minutes of an hour but the final 15 minutes increase to 100 RPM, then the system will reject further requests for up to 15 minutes once the rate limit is reached. After this time expires, requests will be accepted again unless they continue to exceed the rate limit. After being rejected once, the hourly limit will be applied in continuous 15-minute increments until the limit is no longer exceeded within the last 60 minutes.

Burst vs. Sustained Requests

Burst requests are allowed within the rate limits, but keep in mind that you cannot sustain the maximum per-minute rate if the hourly rate does not allow it.

For example, if the rule for an API has limits of 100 requests per minute (RPM) and 2,000 requests per hour (RPH) then the system will allow bursts of up to 100 RPM until the hourly rate limit is reached. In this example, an application or client code could send 100 RPM for up to 20 minutes before being restricted for up to 60 minutes. Since the total hourly rate limit only supports up to 20 minutes of burst spread over the entire hour, it's important not to use the entire hourly rate limit too quickly.

The ability to burst requests may vary by rule, so review both the minute and hourly rate limits before choosing the request rate your application or client code will send.

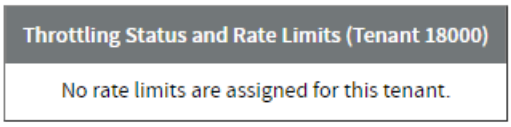
Determining Your Rate Limits

View the exact rate limits for your tenants and their current statuses per API route in the Dev Center, under **Projects** > **Limits**. This page displays one table for all sandboxes associated with your developer account and additional tables for each production tenant. If you are currently operating within the rate limit for a particular API route, then "OK" will be displayed in green. If you exceed the limit and requests are currently being restricted, then "Throttled" will be displayed in red instead.

API Request Limits (Sandboxes)	
/api/platform/{*url} POST, PUT, DELETE 500 requests/min, 10000 requests/hour	OK
/api/platform/{*url} All HTTP methods 500 requests/min, 10000 requests/hour	OK
/api/commerce/catalog/admin/{*url} POST, PUT, DELETE 500 requests/min, 10000 requests/hour	OK
/api/commerce/catalog/admin/{*url} All HTTP methods 500 requests/min, 10000 requests/hour	OK
/api/commerce/inventory/v5/inventory/refresh POST 50 requests/min, 200 requests/hour	OK
/api/commerce/inventory/v5/inventory/adjust POST 50 requests/min, 200 requests/hour	OK
/api/commerce/inventory/{*url} All HTTP methods 500 requests/min, 10000 requests/hour	OK
/api/commerce/{*url} POST, PUT, DELETE 500 requests/min, 10000 requests/hour	OK
/api/commerce/{*url} All HTTP methods 500 requests/min, 10000 requests/hour	OK
/api/{*url} POST, PUT, DELETE 500 requests/min, 10000 requests/hour	OK
/api/{*url} All HTTP methods 500 requests/min, 10000 requests/hour	OK
/{*url} All HTTP methods 500 requests/min, 10000 requests/hour	OK

API Request Limits (Tenant 18005)	
No rate limits are assigned for this tenant.	
API Request Limits (Tenant 18006)	
/api/commerce/catalog/admin/locationinventory/zzzzz/yyyyy All HTTP methods	OK
/api/commerce/catalog/admin/locationinventory/{*url} All HTTP methods	OK
/api/commerce/catalog/admin/locationinventory/{id} All HTTP methods	Throttled
/user/login All HTTP methods	OK
/user/forgotpassword All HTTP methods	OK

If there are no rate limits applied to your developer account, then all sandbox will be "Under Limit" and the production tenant table(s) will say "No rate limits are assigned for this tenant."



Reference the tables below for more details about the rate limits in different environments.

Non-Peak Hours

Certain hours of the day have higher limits in production to support processing jobs and updates when the load is low. For example, requests to Catalog Admin under the /api/commerce/catalog/admin/* route have higher limits overnight during non-peak hours and lower limits during the daytime or peak hours.

US non-peak hours are 5:00 UTC - 11:00 UTC (0:00 CDT - 6:00 CDT, 23:00 CST - 5:00 CST).

EU non-peak hours are 22:00 UTC - 4:00 UTC (0:00 CEST - 6:00 CEST, 23:00 CET - 5:00 CET).

Non-peak hours are calculated in UTC so it is highly recommended to schedule any jobs in UTC and not a local time zone. This will prevent Daylight Saving Time from moving the start and end time.

Rate Limits by Environment

The tables in this section indicate the routes that are currently rate limited per environment. Reference them to determine how to use the Kibo Composable Commerce Platform APIs while staying within the rate limits.

> There may be limits to the physical infrastructure that may restrict the maximum number of requests that can be sent to a given endpoint. Kibo reserves the right to apply additional or different rate limits to ensure platform stability in the case of unreasonable or abusive API activity.

Sandbox

Sandbox rules are applied per developer account and will count requests to all sandboxes. Rate limits will also apply to all sandboxes under the developer account.

Route	HTTP Methods	RPM	RPH	Notes

Route	HTTP Methods	RPM	RPH	Notes
/api/platform/*	POST, PUT, DELETE	500	10000 (166 average RPM)	This does not include dev/app authtickets as they don’t currently support rate limiting.
/api/platform/*	-	500	10000 (166 average RPM)	Excludes POST, PUT, DELETE rule count.
/api/commerce/catalog/admin/*	POST, PUT, DELETE	500	10000 (166 average RPM)	-
/api/commerce/catalog/admin/*	-	500	10000 (166 average RPM)	Excludes POST, PUT, DELETE rule count.
/api/commerce/inventory/v5/inventory/refresh	POST	50	200 (3.33 average RPM)	This API adds to a shared queue so it is limited to avoid backing up the queue for all sandboxes.
/api/commerce/inventory/v5/inventory/adjust	POST	50	200 (3.33 average RPM)	This API adds to a shared queue so it is limited to avoid backing up the queue for all sandboxes.

Route	HTTP Methods	RPM	RPH	Notes
/api/commerce/inventory/*	-	500	10000 (166 average RPM)	Excludes any more specific rules.
/api/commerce/*	POST, PUT, DELETE	500	10000 (166 average RPM)	Excludes any more specific rules.
/api/commerce/*	-	500	10000 (166 average RPM)	Excludes POST, PUT, DELETE rule count and any more specific rules.
/api/*	POST, PUT, DELETE	500	10000 (166 average RPM)	Excludes any more specific rules.
/api/*	-	500	10000 (166 average RPM)	Excludes POST, PUT, DELETE rule count and any more specific rules.
/*	-	500	10000 (166 average RPM)	Storefront and general catch-all rule. Excludes any more specific rules.

Pre-Prod

Pre-Prod currently has very similar rules and rate limits as sandbox. However, pre-prod rules are applied per tenant so one pre-prod tenant will not affect another pre-prod tenant

Route	HTTP Methods	RPM	RPH	Notes
/api/platform/*	POST, PUT, DELETE	500	10000 (166 average RPM)	This does not include dev/app authtickets as they don’t currently support rate limiting.
/api/platform/*	-	500	10000 (166 average RPM)	Excludes POST, PUT, DELETE rule count.
/api/commerce/catalog/admin/*	POST, PUT, DELETE	500	10000 (166 average RPM)	-
/api/commerce/catalog/admin/*	-	500	10000 (166 average RPM)	Excludes POST, PUT, DELETE rule count.
/api/commerce/inventory/*	-	500	10000 (166 average RPM)	Excludes any more specific rules.
/api/commerce/*	POST, PUT, DELETE	500	10000 (166 average RPM)	Excludes any more specific rules.
/api/commerce/*	-	500	10000 (166 average RPM)	Excludes POST, PUT, DELETE rule count and any more specific rules.

Route	HTTP Methods	RPM	RPH	Notes
/api/*	POST, PUT, DELETE	500	10000 (166 average RPM	Excludes any more specific rules.
/api/*	-	500	10000 (166 average RPM)	Excludes POST, PUT, DELETE rule count and any more specific rules.
/*	-	500	10000 (166 average RPM)	Storefront and general catch-all rule. Excludes any more specific rules.

Performance Testing

The Performance Testing environment has a very simple set of rules. When inactive, the limits are very low to not impact any other clients actively using the environment. Please stay within the rate limits and do not do any performance testing or a large number of requests if you are not currently scheduled to use the performance test environment.

Route	HTTP Methods	RPM	RPH	Notes
/api/*	-	Default: 100 Active: 10000	Default: 2000 (33.33 average RPM) Active: 600000 (10000 average RPM)	API catch-all rule.
/*	-	Default: 100 Active: 10000	Default: 2000 (33.33 average RPM) Active: 600000 (10000 average RPM)	Storefront and general catch-all rule. Excludes any more specific rules.

Production

Production tenants are not rate limited. However, to ensure overall platform stability Kibo may apply rate limit rules to a production tenant to limit unreasonable or abusive API activity.

Other Best Practices

Although rate limiting is important, there are additional best practices that can help improve the overall performance of your tenant.

Use Import Export APIs and Analytic Reporting

Leverage the Import/Export APIs for any large-scale transactional data read/write activities. For historical data, leverage the analytic reporting system. The individual REST APIs are intended to support transactional activity and/or asynchronous data synchronization.

Use Bulk APIs

If a bulk API is available, such as for querying or updating inventory, then it is best to use that API instead of submitting individual requests for one item at a time.

However, some bulk APIs have their own restrictions about the amount of data that should be submitted with one request. For example, the Inventory Adjust API only accepts up to 1,000 items per call while the Inventory Refresh API can accept up to 12,000 - however, Kibo recommends making refresh calls with 3,000 items for optimal performance. These limitations are indicated in the API documentation or API-specific context guides where applicable for a specific API.

Use the Right Service

Don't call internal APIs from your Arc actions or applications that aren't designed for it. For example, the Catalog Storefront APIs are designed to support the heavy load of the storefront but the Catalog Administration APIs are not. You should use the appropriate service for your situation when interacting with their Product APIs to ensure a better response time and avoid failures.

Delay Before Retrying

If the responses are not returning the HTTP 200 OK status code (such as HTTP 500 or another error instead), you should add some delay before attempting the request again. Consider using exponential backoff or another strategy instead of retrying immediately.

Limit Response Data

Use the response fields to limit the amount of data returned by API calls, especially when performing queries such as searching for products. Query-string parameters such as page and pageSize (up to a maximum of 200 records per request) can be used to retrieve large amounts of data in digestible chunks. These parameters are always defined in the API documentation for a call that supports them, such as the example Get Products call.

You can also compress the response data to improve API performance when returning large

payloads. Kibo supports the following compression formats:

- Brotli
- Gzip
- Default compression

Send the Accept-Encoding HTTP header with the preferred compression type(s) to enable it for the API response, such as `Accept-Encoding: br, gzip, deflate`. After receiving the response, decompress it to access the full data.

Use Fewer Filters

Though limiting and filtering the response data is useful, it can also be harmful to implement too much complex filter logic. When calling any API with a set of filters, take care to not include too many conditions. While a small set of filters is useful to fine-tune the result set, using a larger number such as 100 "AND" conditions will negatively impact the performance of the query.

Inform Kibo of Expected Traffic

If you have planned promotional activities that may result in a significant increase to server traffic and/or API requests in a short period of time (such as television campaigns, publicity events and sponsorships, email marketing campaigns, or widely advertised sales), Kibo recommends the following best practices:

- Do not send out a major email or text campaign to everyone in a large audience at the exact same time. Instead, space your emails and texts out over a period of time (1-2 hours).
- Send your emails and texts at a time that does not already have high traffic. For instance, most sites see a steady increase in traffic from 7am to 11am Central, so we would recommend that you do not send out an email campaign to several million customers at 8:30am.
- Please read and understand the rest of our best practices for API processes above.

Doing the above will ensure the system can best scale on its own to meet your needs and avoid potential problems. In the event that a large spike or increase in traffic or API requests cannot be avoided, please notify . Include the planned promotional activities, their date/time and duration, and the expected impact such as:

- The reach of the campaign
- The percentage increase of traffic expected from your site
- The expected order volume increase compared to the baseline for your tenant

Support will work with the developer teams to ensure Kibo has the resources in place to support the increase in traffic to your tenant without being surprised by unexpected system load and negatively affecting performance. Note that scaling resources will occur only for your tenant, not

on a site-by-site or regional basis.