LINE SHOPPING API Official Partners Guideline

Prohibiting of LINE SHOPPING API

🤖Prohibition of mass sent requests to the LINE SHOPPING API

- Sending a large number of requests to the LINE SHOPPING API for the purpose of load testing or operation is prohibited.
- Please refer to the specified rate limits that is allowed.

Note

If you send requests exceeding the rate limit, you will receive an error message saying, `429 Too Many Requests`.

Operation tests within rate limits

Even within rate limits, sending the following requests at a high frequency is prohibited. Examples of prohibited action are as follows:

- Repeatedly making requests that aren't for the purpose of using the features provided by the LINE SHOPPING API
- Repeatedly pulling orders/products list without using any filters or conducting hard pull reuqest that overlaps the data.
- Repeatedly retrying on API requests has been responded to with unacceptable retries.

🤖Prohibition of LINE SHOPPING API load tests

There is no service to load test client servers from the LINE SHOPPING API.

For load testing purposes, do not send large numbers of messages via the LINE SHOPPING API. Prepare a separate environment for load testing integration.

🤖Prohibition of requests for non-existent order/product/inventory IDs

---

The return response will be the error `404 Not Found` and may result in penalties.

🤖Prohibition of IP address restrictions

On client servers that receive webhooks, don't restrict access by the IP address of the LINE SHOPPING API from which the webhook request is sent.

We do not disclose any IP address from LINE SHOPPING API and the IP address are also subject to change without notice.

To deny requests from unauthorized sources, use signature validation instead of access control by IP address.

🤖Penalties

When there is an expression that the Company judges to be inappropriate in the LINE SHOPPING API integration, and when there is an act that conflicts with "Prohibitions regarding integration with LINE SHOPPING API" LINE may take the following measures against:

In the case where LINE Company Thailand deems an inappropriate usage or abusive behaviour with the LINE Shopping API integration, which conflicts with "Prohibitions regarding integration with LINE SHOPPING API above points", LINE may take the following actions on your account:

- Termination of MyShop Corporate Agreement
- Block API Key
- Blacklist and Restrict IP Addresses for API Request

In any instance, LINE may not be obliged to provide a reason.

Notes on LINE SHOPPING API integrations

💡 Responses API

Possible response status codes

These HTTP status codes are returned after an API call. We follow the HTTP status code specification unless otherwise stated.

Status Code	Description	Notes
200	`"OK"`	The request was successful.

Status Code	Description	Notes
400	"BAD_REQUEST"	The request could not be understood by the server. The parameter might be invalid.
401	"PERMISSION_DENIED"	The request might contain an invalid API Key.
404	"NOT_FOUND"	The requested resource is not found.
409	"DATA_CONFLICT"	The requested API is in conflict.
429	"TOO_MANY_REQUEST"	The request exceeded the rate limit.
5xx	"INTERNAL_ERROR"	The server could not return the representation due to an internal server error.

For handling error

To handle errors properly, please check the details of the errors.

Error responses body

The following JSON data is returned in the response body when an error occurs.

Response body	Description	Example Value
code String	Error code description. For more details, see Possible response status codes. • "BAD_REQUEST" • "DATA_CONFLICT" • "INTERNAL_ERROR" • "NOT_FOUND" • "PERMISSION_DENIED" • "UNAUTHORIZED"	"BAD_REQUEST"
info Object OPTIONAL	An object of error information. If the object is empty, this object will not be included in the response.	{ "trackingNumber": "tracking no has more than 50 characters" }
message String	Message containing details about the error. For more details, see Details of the error.	"order is invalid"

timestamp Integer	Response timestamp (Unix time).	1646189377206

Details of the error

The details of the error that are found in the `response.body.message` property of the JSON error responses are shown below.

Message	Description
not found	Appears when unable to process an API request by specified ID. Consider these reasons: - Target order ID does not exist - Target product ID does not exist - Target inventory ID does not exist Please try to specify ID correctly.
no Route matched with those values	Appears when calling API that required ID in path parameters. Consider these reasons: - Not specified ID when calling order API such as: o `../orders/{:orderNo}` o `../orders/{:orderNo}/..` - Not specified ID when calling product API such as: o `../products/{:productId}/..` - Not specified ID when calling inventory API such as: o `../inventory/{:inventoryId}/..`
Invalid authentication credentials	Authentication failed when the API was called. Please try to make a new request with the proper API Key.

bad request	Appears when some of the specified request's parameters are incorrect. Consider these reasons: • When calling "Update Tracking Number", but specified `trackingNumber` is more than 50 characters. • When calling "Adjust Inventory", but specified "amount" is incorrect (either lower than the stock on hand or exceeds than maximum stock). • etc. Most of "bad request" will declare which parameters is incorrect in `response.body.info` , please try to correct request's parameters or payload following responses.
order is invalid	Appears when we cannot process a request because the action conflicts logically. Consider these reasons • When calling "Mark as Ship", but order status has already changed status. • When calling "Update Tracking Number", but shipment status is not "Shipped". • etc. Please try to make a new request with follow logical conditions.
internal server error	Appears when something goes wrong, there is an unexpected error or error on the internal server. Please try to make a request again. If it still does not work, please contact customer support.

💡 Rate Limits

A rate limit is the number of API calls that can be made within a given time period. If this limit is exceeded, you will receive an error message saying, **429 Too Many Requests.**

API	Resource	Rate Limit Request		
		Per Hour	Per Minute	Per Second
LINE SHOPPING API	Product API	-	1,000	50
	Order API	-	1,000	50
	Inventory API	-	1,000	50
Other APIs		5,000	-	500

Scope of rate limits

The LINE SHOPPING API applies rate limits for each API group function on a per channel. Note also the following points about the scope of rate limits:

- Even if the endpoint URL is different but within the same API group function, rate limits are applied basing on these considerations.
- Even if your API request receives an error response. We also apply the normal rate limits of API requests.
- We apply rate limits without differentiating between each HTTP method or endpoint URL.
- We apply rate limits without distinguishing between the value of parameters in the URL or the contents of the request body.
- We apply rate limits without distinction, even if you use the endpoint from a different IP address.
- Even if your source URL are the same, we'll apply rate limits independently for each channel.

Response headers

The following HTTP headers are included in LINE SHOPPING API responses:

Response headers	Description	Example Value
X-RateLimit-Remaining-Second `Integer`	Rate limit remaining per `second`. Remaining requests within rate limits per `second`.	499

X-RateLimit-Limit-Second Integer	Rate limit per `second`. Rate limits of API requests per `second`.	500
X-RateLimit-Remaining-Minute Integer	Rate limit remaining per `minute`. Remaining requests within rate limits per `minute`. Note: Rate limit per `minute` will respond following API group's rate limit.	999
X-RateLimit-Limit-Minute Integer	Rate limit per `minute`. Rate limits of API requests per `minute`. Note: Rate limit per `minute` will respond following API group's rate limit.	1000
RateLimit-Reset Integer	Additional headers to show the time remaining (in seconds) until the quota is reset	1
RateLimit-Remaining Integer	Additional headers to show the number of available requests.	499
RateLimit-Limit Integer	Additional headers to show the allowed limits.	500
date Date	Request date. The date/time of request was received. (UTC Time)	1 Nov 2022 14:05:30 GMT
x-line-oap-request-id String	Request ID. An ID is issued for each request.	5ba1a2ca-5666-4a8f-ba4a-4142194a6af3

For more detail about Rate Limit in header field

To understand how RateLimit-Limit, RateLimit-Remaining, and RateLimit-Reset are defined clearly, please see this document Rate Limit Definitions.

For "x-line-oap-request-id"

Please make notes that "x-line-oap-request-id" must be provided when an issue has occurred for use to identifying the case; therefore you must keep this when you received a response for the purpose of backtracking.

The headers RateLimit-Limit, RateLimit-Remaining, and RateLimit-Reset are referenced and referred from RateLimit Header Fields for HTTP. These may change if the specification is updated.

💡 Webhooks

When an event occurs, such as changing the order status, the LINE SHOPPING API sends an HTTPS request to the webhook URL (client-server).

The webhook URL is configured in the Shop settings > OPEN API, or check the Webhook URL setting instruction.

We recommend that you make the event processing asynchronous

We recommend that you make the event processing asynchronous so that the processing of HTTP requests does not delay the processing of subsequent events.

The IP address of the LINE SHOPPING API isn't disclosed

The IP address of the LINE SHOPPING API from which the webhook request is sent to is not disclosed. For better security, use signature validation instead of access control by IP address.

Webhook URL validation rules

Ensure these webhook URL validation rules are met

- Enter a valid HTTPS URL.
- Must be 500 characters or less.

Webhook request headers

Request headers	Description
x-myshop-signature any	Used for LINE SHOPPING API signature validation.
x-oap-request-id any	Used for tracing each request ID.

Request header field names are case insensitive

Uppercase and lowercase letters in the Request headers field names may change without notice. The client server that receives the webhook should handle the header field name without case distinction.

	Before change	After change
Header field name example	X-MyShop-Signature	x-myshop-signature

For more detail, please check https://datatracker.ietf.org/doc/html/rfc7230#section-3.2.

Webhook request body

The request body contains a JSON object with the webhook details and an object of the webhook event.

Request body	Description
event Object	An object of webhook event, which will be included. • name String : An event name • timestamp String : Timestamp of the event

To see the details of each webhook event request body

For checking all of the request body when LINE SHOPPING API sends webhook, please check Webhook API specification.

Response webhook request

The client-server must return status code `200` after it receives the HTTP POST request sent from the LINE SHOPPING API.

Note

- Even if the client-server fails to receive the HTTP POST request sent from the LINE SHOPPING API, the client-server can receive this request again by webhook retry. For more information, see Webhook Retry Logic.
- If the client-server does NOT respond within 5 seconds after receiving a request, LINE SHOPPING API will close the request and consider that request a failure.

Security warning

Your client-server may receive malicious HTTP POST requests that were not sent from the LINE SHOPPING API.

Do not process any webhook event objects until you have verified their signatures.

Webhook Retry Logic

The LINE SHOPPING API provides a feature to retry webhooks that are not received on the client server. Even if the client-server fails to respond normally to a webhook due to temporary over-access or other reasons, the webhook will be retried from LINE SHOPPING API for a certain period of time so that the webhook can be received after the client-server has recovered.

Note

Our system treats 200 status and responses within 5 seconds as a successful request received. Any other response to our system is considered to be a failure. Our system retry mechanism is to send webhook event 9 times in an hour interval by exponential backoff.

Webhook signature validation

The signature in the x-myshop-signature request header must be verified to confirm that the request was sent from LINE SHOPPING API.
Authentication is performed as follows:

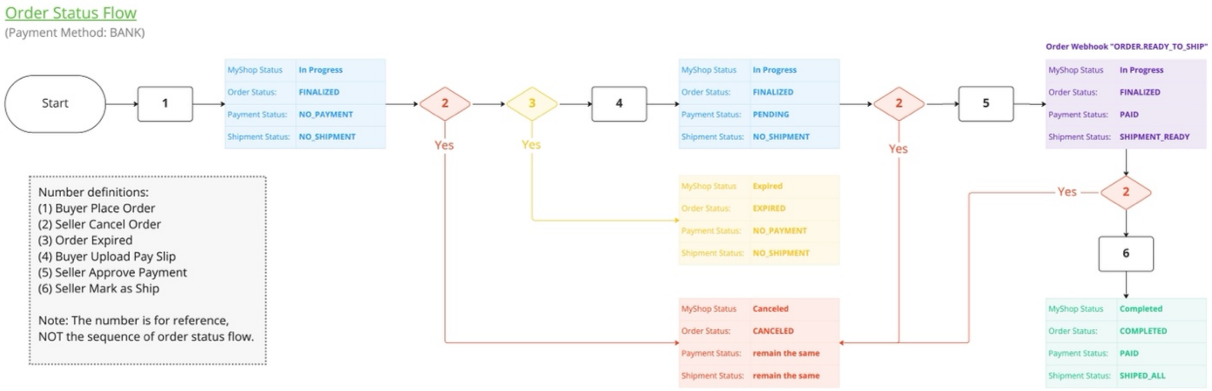
- With the token as the secret key, your application retrieves the digest value in the request body created using the HMAC-SHA256 algorithm.
- The server confirms that the signature in the request header matches the digest value, which is Base64 encoded.

More details about the example code of signature validation

Please see verifying signatures for more details about signature validation and example code.

💡 LINE SHOPPING API Order Status Flow

Order Status Flow (Payment Method: BANK)



MyShop Status	In Progress
Order Status:	FINALIZED
Payment Status:	NO_PAYMENT
Shipment Status:	NO_SHIPMENT

MyShop Status	In Progress
Order Status:	FINALIZED
Payment Status:	PENDING
Shipment Status:	NO_SHIPMENT

MyShop Status	In Progress
Order Status:	FINALIZED
Payment Status:	PAID
Shipment Status:	SHIPMENT_READY

MyShop Status	Expired
Order Status:	EXPIRED
Payment Status:	NO_PAYMENT
Shipment Status:	NO_SHIPMENT

MyShop Status	Canceled
Order Status:	CANCELED
Payment Status:	remain the same
Shipment Status:	remain the same

MyShop Status	Completed
Order Status:	COMPLETED
Payment Status:	PAID
Shipment Status:	SHIPED_ALL

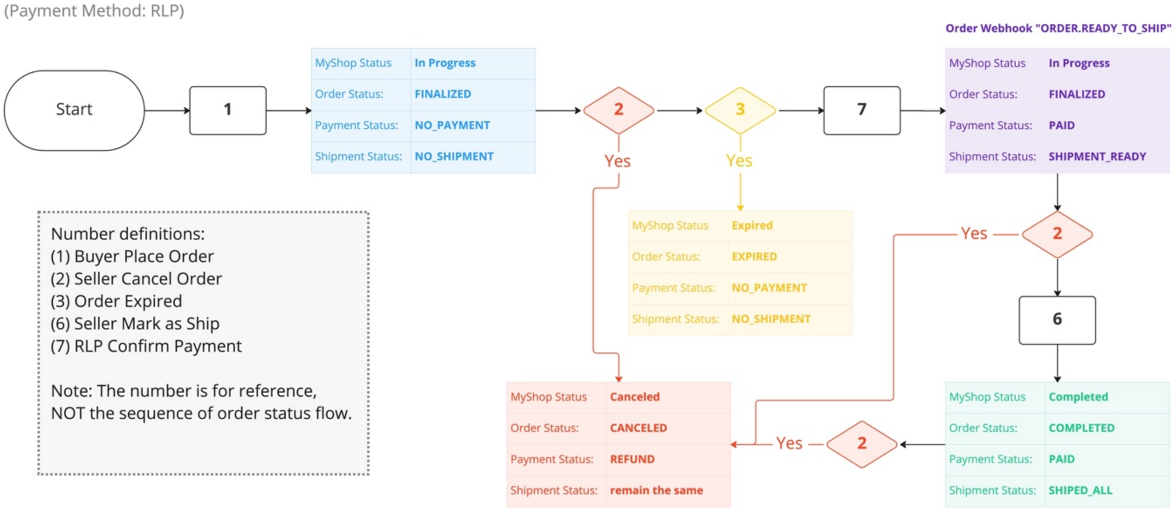
Number definitions:
(1) Buyer Place Order
(2) Seller Cancel Order
(3) Order Expired
(4) Buyer Upload Pay Slip
(5) Seller Approve Payment
(6) Seller Mark as Ship

Note: The number is for reference, NOT the sequence of order status flow.

Order Status Flow (Payment Method: RLP)

Order Status Flow

(Payment Method: RLP)



MyShop Status	In Progress
Order Status:	FINALIZED
Payment Status:	NO_PAYMENT
Shipment Status:	NO_SHIPMENT

MyShop Status	Expired
Order Status:	EXPIRED
Payment Status:	NO_PAYMENT
Shipment Status:	NO_SHIPMENT

MyShop Status	In Progress
Order Status:	FINALIZED
Payment Status:	PAID
Shipment Status:	SHIPMENT_READY

MyShop Status	Canceled
Order Status:	CANCELED
Payment Status:	REFUND
Shipment Status:	remain the same

MyShop Status	Completed
Order Status:	COMPLETED
Payment Status:	PAID
Shipment Status:	SHIPED_ALL

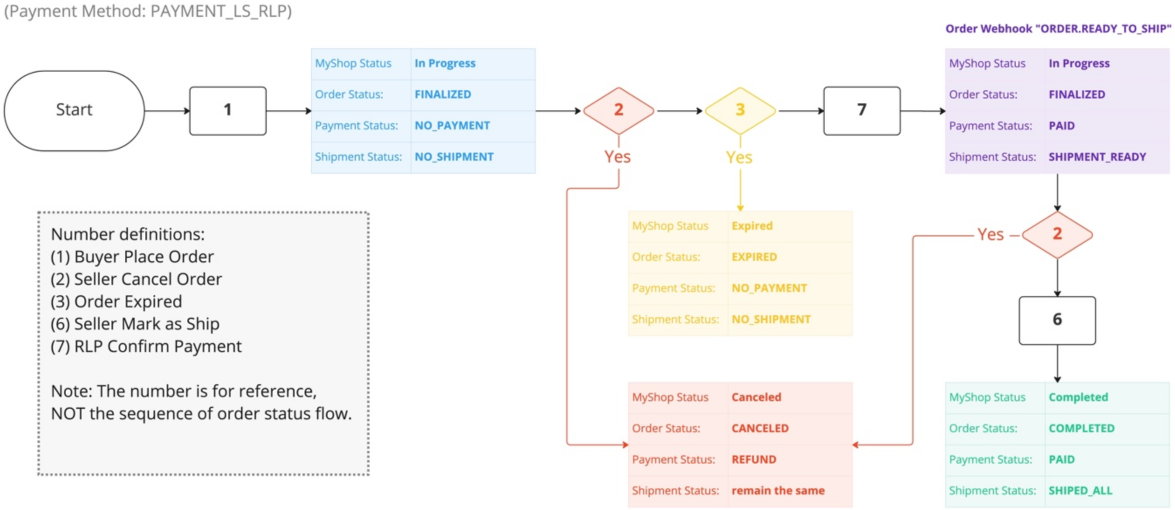
Number definitions:
(1) Buyer Place Order
(2) Seller Cancel Order
(3) Order Expired
(6) Seller Mark as Ship
(7) RLP Confirm Payment

Note: The number is for reference, NOT the sequence of order status flow.

Order Status Flow (Payment Method: PAYMENT_LS_RLP)

Order Status Flow

(Payment Method: PAYMENT_LS_RLP)



MyShop Status	In Progress
Order Status:	FINALIZED
Payment Status:	NO_PAYMENT
Shipment Status:	NO_SHIPMENT

MyShop Status	Expired
Order Status:	EXPIRED
Payment Status:	NO_PAYMENT
Shipment Status:	NO_SHIPMENT

MyShop Status	In Progress
Order Status:	FINALIZED
Payment Status:	PAID
Shipment Status:	SHIPMENT_READY

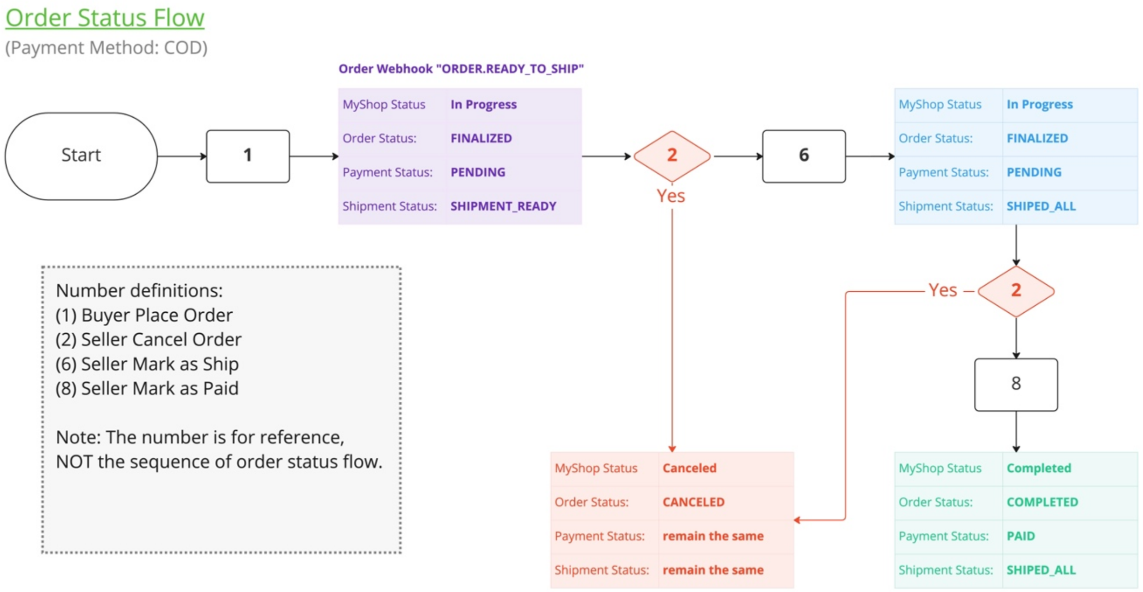
MyShop Status	Canceled
Order Status:	CANCELED
Payment Status:	REFUND
Shipment Status:	remain the same

MyShop Status	Completed
Order Status:	COMPLETED
Payment Status:	PAID
Shipment Status:	SHIPED_ALL

Number definitions:
(1) Buyer Place Order
(2) Seller Cancel Order
(3) Order Expired
(6) Seller Mark as Ship
(7) RLP Confirm Payment

Note: The number is for reference, NOT the sequence of order status flow.

Order Status Flow (Payment Method: COD)



MyShop Status	In Progress
Order Status:	FINALIZED
Payment Status:	PENDING
Shipment Status:	SHIPMENT_READY

MyShop Status	In Progress
Order Status:	FINALIZED
Payment Status:	PENDING
Shipment Status:	SHIPED_ALL

MyShop Status	Canceled
Order Status:	CANCELED
Payment Status:	remain the same
Shipment Status:	remain the same

MyShop Status	Completed
Order Status:	COMPLETED
Payment Status:	PAID
Shipment Status:	SHIPED_ALL

Number definitions:
(1) Buyer Place Order
(2) Seller Cancel Order
(6) Seller Mark as Ship
(8) Seller Mark as Paid

Note: The number is for reference, NOT the sequence of order status flow.

Order Status Data Definitions

Order status details

Order Status	Note
FINALIZED	The cart has placed an order.
COMPLETED	Seller marks an order as shipped.
EXPIRED	Buyer didn't complete payment within the duration of time. (It depends on the Seller's setting, default = 3 days)
CANCELED	Seller cancels the order.

Payment method details

Payment method	Note
RLP	Rabbit LINE Pay
COD	Cash on delivery
BANK	Bank transfer

Payment method	Note
PAYMENT_LS_RLP	New Payment RLP

Payment status details

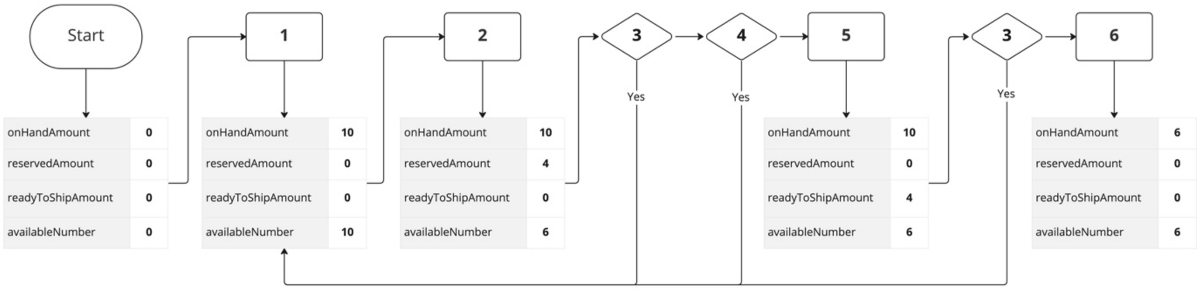
Payment status	Note
NO_PAYMENT	-
PENDING	Waiting for payment.
PAID	An order has been paid.
REFUND	An order has been refunded.

Shipment status details

Shipment status	Note
NO_SHIPMENT	-
PENDING	Shipping address is waiting to be confirmed (Gift order only)
SHIPPED_ALL	An order has been marked as shipped.
SHIPMENT_READY	An order is ready for shipment.

💡 LINE SHOPPING API Inventory Flow

Inventory Flow



Number definitions:
(1) Seller Add Stock 10 Qty.
(2) Buyer Place Order
(3) Seller Cancel Order
(4) Order Expired
(5) Buyer Complete Payment
(6) Seller Mark as Ship

Note: The number is for reference,
NOT the sequence of order status flow.

Inventory Flow Data Definition

Shipment status	**Note**
onHandAmount	Quantity in stock (Include ‘Reserved’ and ‘Ready to ship’ status)
reservedAmount	Quantity that customers do not complete payment
readyToShipAmount	Quantity that customers complete payment and wait to ship
availableNumber	Quantity that is available and displayed on the storefront (Exclude ‘Reserved’ and ‘Ready to ship’ status)

Implement Suggestions when integrating with LINE SHOPPING API

👍Saving logs

We recommend saving logs for LINE SHOPPING API requests and webhooks received for a certain period of time so that developers themselves can smoothly investigate the cause and scope of a problem when it occurs. Because we cannot be provided a complete log when requests had never arrived on our server (eg. Request timeout) and also we keep a log for a limited time for auditing purposes but it can not provide a complete trail without a request log from the source.

Logs for LINE SHOPPING API request

We recommend saving the following information as a log when making a request to the LINE SHOPPING API.

- Request ID (`x-line-oap-request-id`) of a Response header
- Time of API request
- Request method
- Request URL
- Status code returned in response by the LINE SHOPPING API

More specifically, save it in a log file using the following format

Request ID	**Time of API Request**	**Request method**	**Request URL**	**Response code**
5ba1a2ca-5666-4a8f-ba4a-4142194a6af3	1 Nov 2022 14:05:30	POST	https://developers-oaplus.line.biz/myshop/v1/products	200

Additional information that would be useful to keep in log for API Request

Depending on the integration requirements, the following information, in addition to the above, can be stored for investigation when problems occur.

- Response body returned by the LINE SHOPPING API after the API request

We reserve the right to provide logs of LINE SHOPPING API request

We do not have any obligations to provide logs of LINE SHOPPING API requests from the client-server, etc., despite inquiries. Logs should be saved by the developers themselves.

Logs for webhook received

We recommend saving the following information as a log when you receive a webhook from the LINE SHOPPING API through the client-server. Also, for the backup source of webhook logs, we provide logging in OpenAPI setting for 60 days.

- Request ID (`x-line-oap-request-id`) of the Request header
- Time webhook was received
- Response body returned by the client-server after the received webhook

More specifically, save it in a log file using the following format.

Request ID	Time webhook was received	Response Body

5ba1a2ca-5666-4a8f-ba4a-4142194a6af3	1 Nov 2022 14:05:30	{
		"success": true,
		"timestamp": "2020-09-30T05:38:20.031Z",
		"statusCode": 200,
		"reason": "OK",
		"detail": "200"
		}

Additional information that would be useful to keep in log for webhook received

Depending on the integration requirements, the following information, in addition to the above, can be stored for investigation when problems occur.

- Webhook event object sent by LINE SHOPPING API
- Signature (`x-myshop-signature`) of the request header when webhook is sent from the LINE SHOPPING API

👍Retry API method

Even when there's no LINE SHOPPING API failure, these problems may occur due to the network connection status of the client-server and other factors.

- API requests did not complete successfully
- Can not get a response from the LINE SHOPPING API properly

Retry API requests as needed

Design how to retry an API request according to the response status as follows.

Status code	Description	Need to retry or not
500 Internal Server Error	Internal server error.	✅ The request may be successful by retrying.
504 Gateway Timeout	The request failed due to a network failure or some other reason.	✅ The request may be successful by retrying.
200 status code	The API request has been accepted.	❌ Additional retries will not be accepted.
409 Conflict	An API request conflict	❌ The request has been a conflict
429 Too many requests	The request is exceeded the rate limit.	❌ Should stop making additional API requests until enough time has passed to retry
4xx	Problem with the request.	❌ Trying again does not change the results.

Additional API requests before retrying

Even retry any API request following the response code suggestion, you should consider the additional API request before retrying failure. For more details please check the Additional API requests before retrying if failed section.

The interval between retries

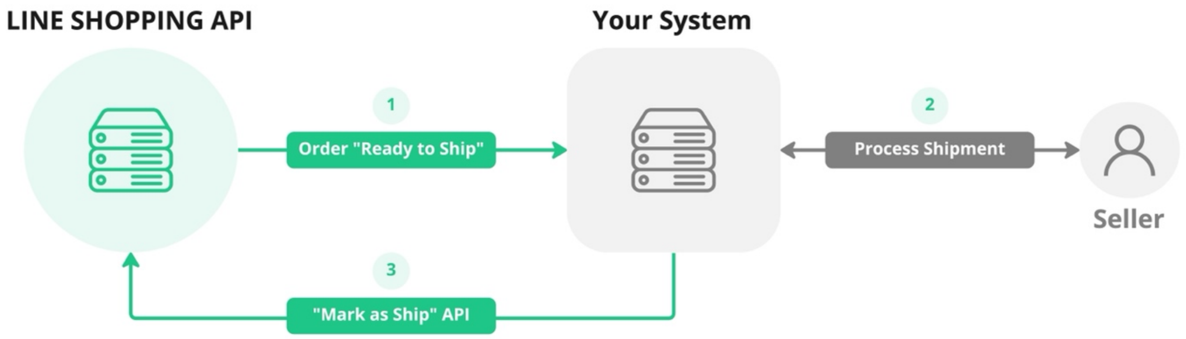
- Since any API requests are counted in the API request count, even if your API request receives an error response, frequent retries may cause the API rate limit to be reached.
- We recommend you control the retry interval by exponential backoff.

Additional API requests before retrying if failed

On some API requests, it may succeed even return an error or a connection was lost during the request. And if you try to retry the request without additional requesting before, the results may conflict because previous requests were successful, so we recommend that you use additional API requests before trying failed requests again.

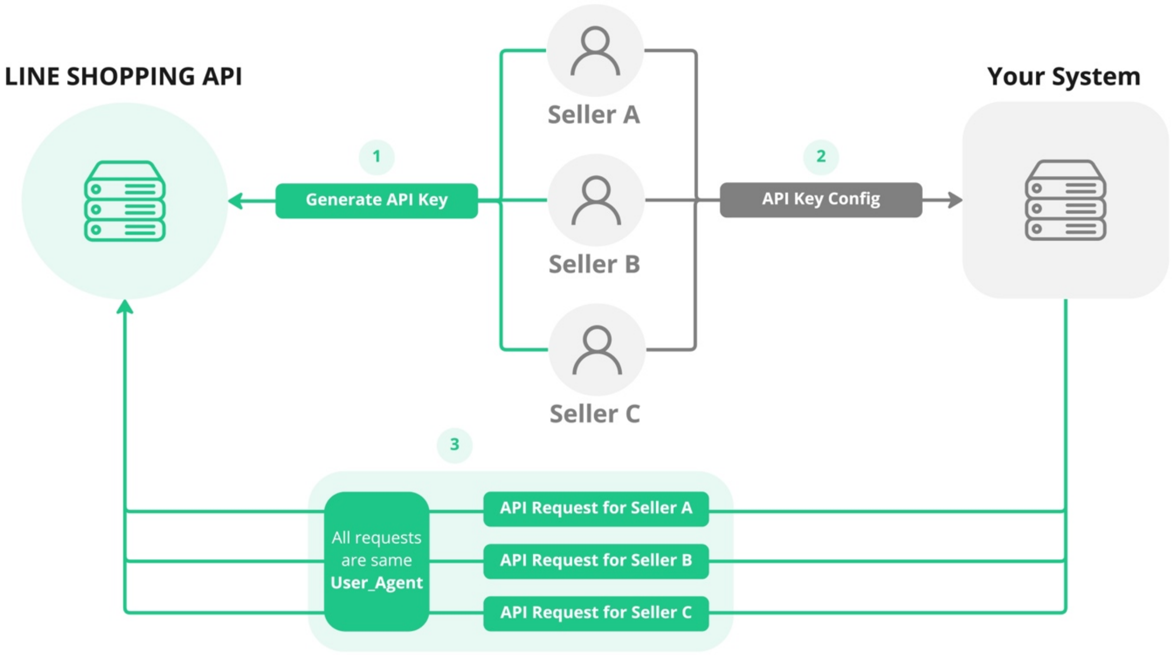
API Group	API Name	HTTP Method	URL	Suggestion before retrying
Inventory	Decrease Inventory	PUT	/inventory/{id}/decrease	Should call "Get all products" to check and update the current number of stock before retrying again.
	Increase Inventory	PUT	/inventory/{id}/increase	Should call "Get all products" to check and update the current number of stock before retrying again.
Order	Mark as paid (COD)	POST	/orders/{orderNo}/mark-as-paid	Should call "Get all orders" or "Order details" to check and update the current order status before retrying again.
	Mark as ship	POST	/orders/{orderNo}/mark-as-ship	Should call "Get all orders" or "Order details" to check and update the current order status before retrying again.

👍Always Update the Order Status to LINE SHOPPING API



When you get an order list to your system by API request and the Seller completes shipment on your platform, please always update the order status to LINE SHOPPING API by "Mark as Ship" API. If your system NOT updating the order status and the Seller also did not "Mark as Ship" order in LINE SHOPPING platform, the order will still be on the status "Ready to Ship", which means the order will remain in the system until having any update. For more details about the "Mark as Ship" API, please see on LINE SHOPPING API document.

👍Always specify `User_Agent` in the Request Header by correctly



To be easier to identify requestors who make API requests to LINE SHOPPING API, in addition to OpenAPI Key. We suggest you specify `User_Agent` to your request header for tracking and tracing back.

In case your platform has many sellers enabled to connect between your platform and LINE SHOPPING API, that will be extra benefits on checking in the overall integration by we can identify that all of the requests from sellers who specify the same `User_Agent` in request's header are from your platform.

Header	Description	Notes
User-Agent	The User-Agent request header is a character string that lets servers and network peers identify the application, operating system, vendor, and/or version of the requesting user agent. Suggestion value is your `service name`, or `company name`.	`User-Agent: LINE SHOPPING Platform`

👍Optimize API Getting Order List Requests with Specific Order Status

To get an orders list by LINE SHOPPING API, we provide an API that can get orders with specified query parameters to prevent overlapping and unnecessary orders. Please see LINE SHOPPING API documents for more information about getting all orders API specifications.

End-Point URL	/myshop/v1/orders
HTTP Method	GET

Order Status Flow

For more details about the order status flow, please see LINE SHOPPING API Order Status Flow.

Get Orders whose Payment Status is "Pending"

After orders have been placed, if the Buyer NOT selected the shipment method as "COD", the payment status will be changed to "Pending", which is the order pending to let the Buyer complete their payment either "Bank transfer" or "Rabbit LINE Pay" or "New Payment".

Normally, this is the first stage of LINE SHOPPING to get any orders which been placed from LINE SHOPPING platform; except the order which selected the shipment method is "COD", which will be changed stage to "READY_TO_SHIP".

The proposed way to send query parameters to get an order's payment status is "Pending", as below.

Query Parameters	Values
`orderStatus`	`"FINALIZED"`
`paymentStatus`	`"PENDING"`

Get Orders whose Shipment Status is "Ready to Ship"

After the order is completed the payment, the shipment status will be changed to "READY_TO_SHIP", which is the order being ready to let the Seller ship the order to the Buyer.

The proposed way to send query parameters to get an order's shipment status is "Ready to Ship", as below.

Query Parameters	Values
`orderStatus`	`"FINALIZED"`
`shipmentStatus`	`"SHIPMENT_READY"`

Note

If you need orders whose shipment status is "Ready to Ship", we suggest you use Order Webhook "ORDER.READY_TO_SHIP".

Get Orders whose Order Status is "Expired"

After the Buyer places an order to LINE SHOPPING platform but has NOT completed payment within the specified order expiration date in the setting, the order will change status to "Expired", which may need updating status on your system when you pull the order that payment status is "Pending" previously.

The proposed way to send query parameters to get an order status is "Expired", as below.

Query Parameters	Values
`orderStatus`	`"EXPIRED"`
`paymentStatus`	`"NO_PAYMENT"`
`shipmentStatus`	`"NO_SHIPMENT"`

Get Orders whose Order Status is "Canceled"

When the Seller cancels an order that was placed to LINE SHOPPING platform, the order will change status to "Canceled", which may need updating status on your system when you pull the order previously.

The proposed way to send query parameters to get an order status is "Canceled", as below.

Query Parameters	Values
`orderStatus`	`"CANCELED"`

Using `startAt` and `endAt` to scope query order list

To scope the order list, we provide query parameters `startAt` and `endAt` to set a range of order lists that will help with query performance and avoid query overlapping data.

Query Parameters	Values
`startAt`	Your any specific start date of your query range you preferred or your last query's `endAt` period to NOT get duplicated data.

`endAt`	Your end of the range you preferred or your next interval to get the order list period.

Note

- Try to prevent sending overlapping `startAt` and `endAt` period to avoid overlapping data
- `startAt` and `endAt` should NOT be long-range because that affects performance

Additional Proposed Way to get orders which shipment status "Ready to Ship" by using Order Webhook "ORDER.READY_TO_SHIP"

After you complete the webhook API setting, when the order status has been changed and the event "READY_TO_SHIP" is triggered, we will send a webhook to the client-server with an order model payload to update the order status on your system without any request required. To read more details about the payload, please check on LINE SHOPPING API document.

Event Name	`"ORDER.READY_TO_SHIP"`

Sending webhook API when Order-, Payments, Shipment Status follows conditions as below.

Payment Method	Order Status	Payment Status	Shipment Status
BANK or RLP or PAYMENT_LS_RLP	`FINALIZED`	`PAID`	`SHIPMENT_READY`
COD	`FINALIZED`	`PENDING`	`SHIPMENT_READY`