Kitaboo Order & Reader Launch API Guide

Audience: Client integrators (LMS, e-store, custom portals)
Services: `DistributionServices` (Backend-Rest) + `backend-micro-render` (`micro-lti`) + `micro-authorization`
Last updated: 2026-09-22
Supersedes: *Copy of Order API Document* (Word/HTML export)

Table of contents

---

1. Integration flows

All **order** operations use the same Rest OAuth token (§3). **Launch** uses a separate microservices JWT (§5). Store your own `orderNo` for every create — you need it for update and delete.

Field alignment (order ↔ launch)

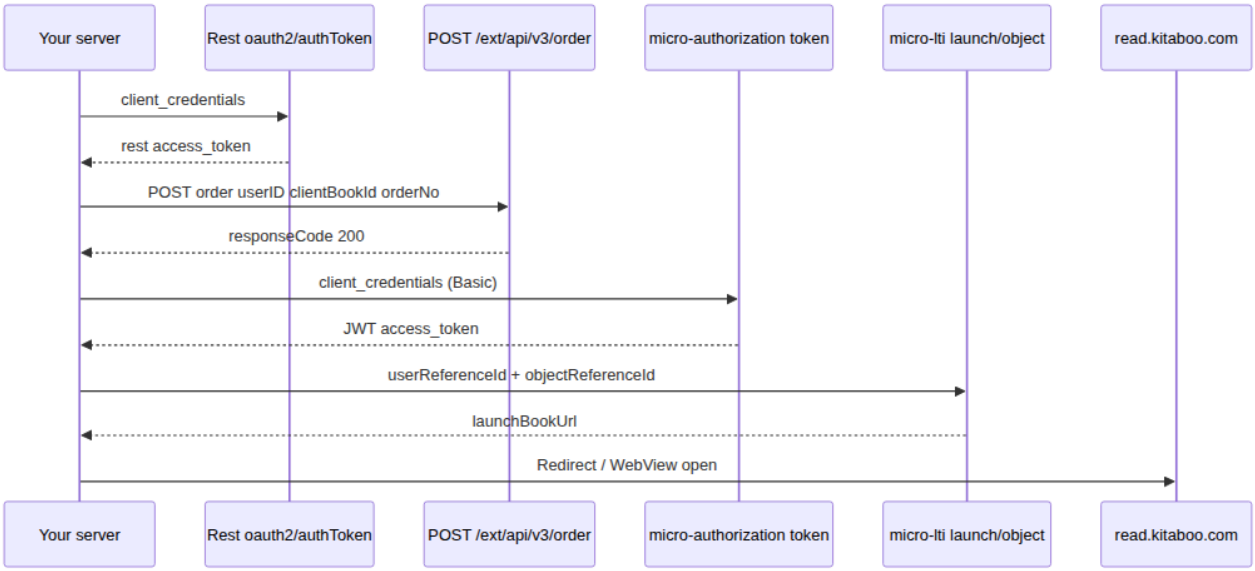
Order API (Rest)	Launch API (`micro-lti`)	Meaning
`userID`	`userReferenceId`	Your stable user identifier
`clientBookId`	`objectReferenceId`	Your book/content reference (omit on launch to open **bookshelf** only)

Order API (Rest)	Launch API (`micro-lti`)	Meaning
`orderNo`	—	Your order reference; required for PUT/DELETE, not used at launch

---

1.1 Assign book and open reader (full journey)

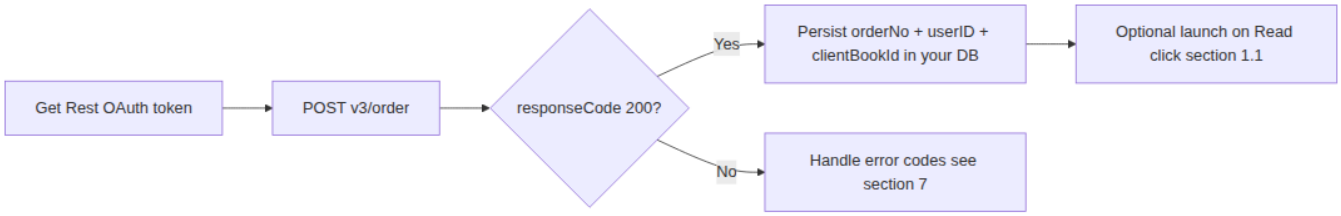
Use when a user purchases or is entitled to a title and should **read immediately** in Kitaboo.



Step	API	Details
1	`POST {rest-host}/oauth2/authToken`	Rest Bearer for order calls (§3).
2	`POST {rest-host}/ext/api/v3/order`	Assign book; see §4.1.
3	`POST {gateway}/v1/micro-authorization/oauth/token`	Gateway JWT (§5).
4	`POST {gateway}/v1/micro-lti/auth/launch/object`	Bookshelf or single book (§6).
5	Browser / WebView	Load `launchBookUrl` (short-lived; do not cache long-term).

1.2 Assign book to user (create order only)

Use when entitlement is created on your side (checkout, LMS enrollment) and the user will open Kitaboo **later**, or launch is handled by another channel.



Steps

1. **Authenticate** — `POST {rest-host}/oauth2/authToken` with `grant_type=client_credentials` (§3).
2. **Create order** — `POST {rest-host}/ext/api/v3/order` with at minimum:

Field	Notes
`orderNo`	Unique per purchase/assignment in **your** system
`userID`	Becomes `userReferenceId` at launch
`clientBookId`	Becomes `objectReferenceId` at launch
`firstName`, `lastName`, `userName`, `password`, `email`	User profile / login
`endDate`	Optional `yyyy-MM-dd`; omit for perpetual
`type`	`1` individual, `5` bulk (per contract)

1. **Confirm** — JSON `responseCode` `200` and `responseMsg` e.g. *Order created successfully* (§4.1).
2. **Idempotency** — Re-posting the same `orderNo` may return duplicate errors (`106`); use a new `orderNo` or switch to update flow if extending the same commercial order.

Minimal curl sequence

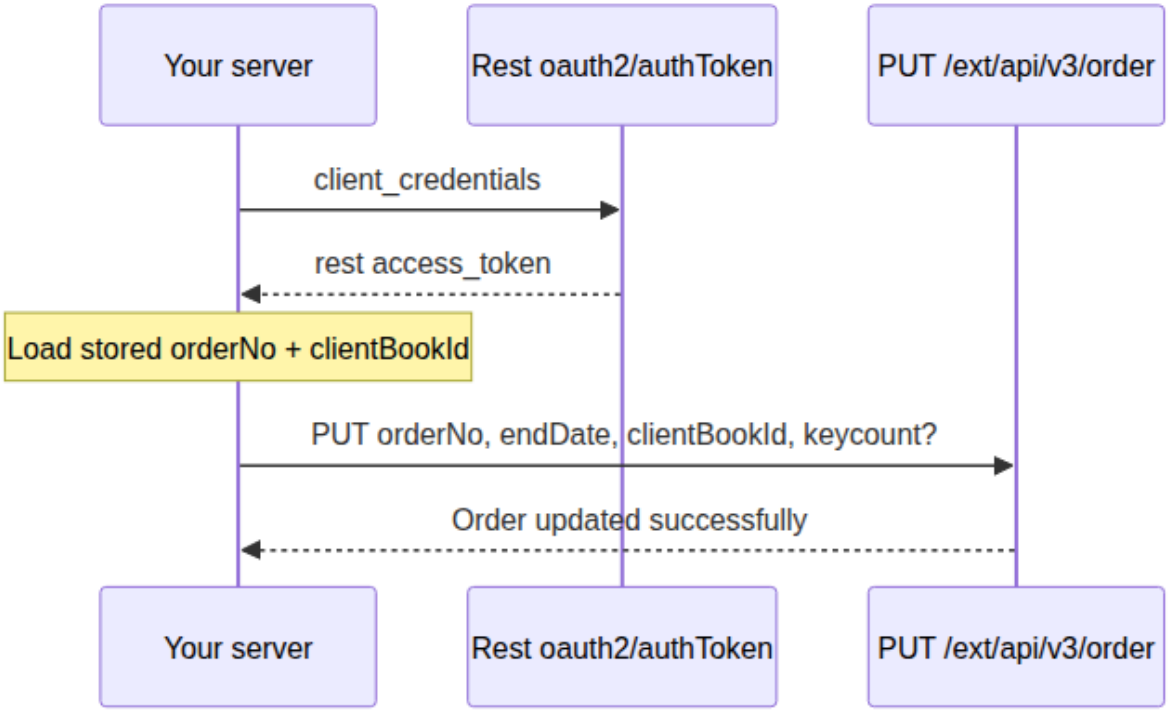
```
# 1) Token
REST_ACCESS_TOKEN=$(curl -sS -X POST "${REST_HOST}/oauth2/authToken" \
  -H "Content-Type: application/x-www-form-urlencoded" \
  --data-urlencode "grant_type=client_credentials" \
  --data-urlencode "client_id=${KITABOO_CLIENT_ID}" \
  --data-urlencode "client_secret=${KITABOO_CLIENT_SECRET}" \
  | jq -r '.access_token')

# 2) Assign book
curl -sS -X POST "${REST_HOST}/ext/api/v3/order" \
  -H "Authorization: Bearer ${REST_ACCESS_TOKEN}" \
  -H "Content-Type: application/x-www-form-urlencoded" \
  --data-urlencode "orderNo=ORD-2026-001" \
  --data-urlencode "clientBookId=9780573800085" \
  --data-urlencode "userID=demo-user-005" \
  --data-urlencode "firstName=Demo" \
```

```
  --data-urlencode "lastName=User" \
  --data-urlencode "userName=demo005@example.com" \
  --data-urlencode "password=dummy" \
  --data-urlencode "email=demo005@example.com"
```

1.3 Update order end date

Use to **extend**, **shorten**, or set expiry on an existing license (renewal, subscription renewal, grace period). User and book are already tied to the order via `orderNo` and `clientBookId`.



Steps

1. **Authenticate** — same Rest OAuth as §3 (token may be reused until `expires_in`).
2. **Resolve identifiers** — use the **same** `orderNo` sent at create time; `clientBookId` must match the licensed title.
3. **Update** — `PUT {rest-host}/ext/api/v3/order`:

Field	Required	Purpose
`orderNo`	Yes	Your order reference
`clientBookId`	Yes	Book reference for that order
`endDate`	No*	New expiry `yyyy-MM-dd`; omit only if your contract allows perpetual via update
`keycount`	No	Seat/license count for group orders

*To move to perpetual, confirm with Kitaboo support — behavior may depend on client configuration.

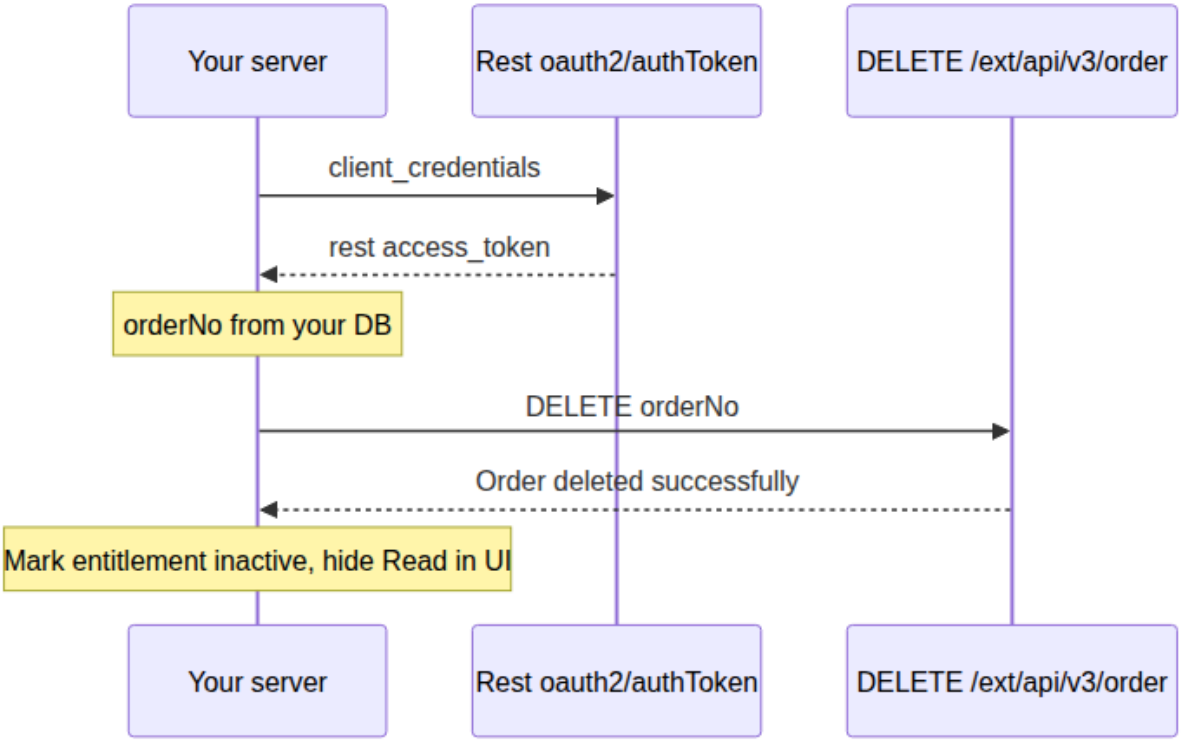
1. **Confirm** — `responseCode` `200`, `responseMsg` *Order updated successfully* (§4.2).
2. **Reader access** — no launch call required; existing sessions may need re-login to pick up new expiry.

Example (extend to 2028-08-20)

```
curl -sS -X PUT "${REST_HOST}/ext/api/v3/order" \
  -H "Authorization: Bearer ${REST_ACCESS_TOKEN}" \
  -H "Content-Type: application/x-www-form-urlencoded" \
  --data-urlencode "orderNo=ORD-2026-001" \
  --data-urlencode "clientBookId=9780573800085" \
  --data-urlencode "endDate=2028-08-20"
```

1.4 Delete order

Use when a purchase is **refunded**, enrollment is **revoked**, or access must be **removed**. After delete, launch for that title should fail until a new order is created.



Steps

1. **Authenticate** — Rest OAuth (§3).
2. **Delete** — `DELETE {rest-host}/ext/api/v3/order` with form body **`orderNo`** only (§4.3).
3. **Confirm** — `responseCode` `200`, `responseMsg` *Order deleted successfully*.

4. **Downstream** — stop offering “Read” for that `orderNo`; create a **new** `orderNo` if the user buys again (do not assume delete is reversible).

Example

```
curl -sS -X DELETE "${REST_HOST}/ext/api/v3/order" \
  -H "Authorization: Bearer ${REST_ACCESS_TOKEN}" \
  -H "Content-Type: application/x-www-form-urlencoded" \
  --data-urlencode "orderNo=ORD-2026-001"
```

1.5 Flow comparison

Goal	Rest OAuth	Order method	Launch (micro-lti)	Keep in your DB
Assign book	Yes	`POST` v3/order	Optional now; required to open reader	`orderNo`, `userID`, `clientBookId`
Extend / change expiry	Yes	`PUT` v3/order	No	Same `orderNo`
Revoke access	Yes	`DELETE` v3/order	No (entitlement removed)	Archive `orderNo`
Open bookshelf	MS JWT only*	—	`POST` launch without `objectReferenceId`	`userID`
Open one book	MS JWT only*	Order must exist	`POST` launch with `objectReferenceId`	`userID`, `clientBookId`

*User must already have entitlement from a prior **POST** order unless your tenant allows launch-only test users.

Suggested integration order for new clients

1. Implement **§1.2** (assign) and verify `200` in QA.
2. Implement **§1.3** and **§1.4** against test `orderNo` values.
3. Add **§1.1** launch once orders are stable.
4. Wire storefront/LMS buttons: *Purchase* → POST order; *Renew* → PUT; *Refund* → DELETE; *Read* → launch.

---

2. Environments

DistributionServices (Order API + Rest OAuth)

Region	Base URL (deploy as `{rest-host}`)
US production	`https://cloud-us.kitaboo.com/DistributionServices`
EU production	`https://cloud.kitaboo.eu/DistributionServices`
Kitaboo One production	`https://cloud.kitaboo.in/DistributionServices`
Staging / QA	`https://qacloud.kitaboo.com/DistributionServices`

External API path prefix: `/ext/api/`
Example order endpoint: `{rest-host}/ext/api/v3/order`

Microservices gateway (token + launch)

Region	API gateway
US	`https://microservices.kitaboo.com`
EU	`https://microservices.kitaboo.eu`
India	`https://microservices.kitaboo.in`
Staging	`https://stagmicro.kitaboo.com`

API	Path
OAuth token	`/v1/micro-authorization/oauth/token`
Launch bookshelf / book	`/v1/micro-lti/auth/launch/object`

---

3. Authentication — DistributionServices (Order API)

Kitaboo supports **OAuth 2.0 client credentials** for server-to-server calls to `DistributionServices`.

Request

Item	Value
Method	`POST`
URL	`{rest-host}/oauth2/authToken`
Content-Type	`application/x-www-form-urlencoded`

Body parameters

Parameter	Type	Required	Description
`grant_type`	string	Yes	`client_credentials`
`client_id`	string	Yes	OAuth consumer key for your Kitaboo client
`client_secret`	string	Yes	OAuth secret

Example (curl)

```
REST_HOST="https://cloud-us.kitaboo.com/DistributionServices"

curl -sS -X POST "${REST_HOST}/oauth2/authToken" \
  -H "Content-Type: application/x-www-form-urlencoded" \
  --data-urlencode "grant_type=client_credentials" \
  --data-urlencode "client_id=${KITABOO_CLIENT_ID}" \
  --data-urlencode "client_secret=${KITABOO_CLIENT_SECRET}"
```

Sample response

```
{
  "expires_in": 3600,
  "access_token": "43c3e0cae11b4007b0c5da8abe27f402"
}
```

Using the token on Order API

```
Authorization: Bearer {access_token}
Content-Type: application/x-www-form-urlencoded
```

---

4. Order API (Rest / External API)

The Order API creates or updates **user + license** data in Kitaboo (who can read which title, and until when). It is used from LMS, storefronts, and integration middleware.

Base path: `{rest-host}/ext/api/v3/order`

4.1 Create order — assign book to user

Item	Value
Method	`POST`
Auth	`Authorization: Bearer {rest_access_token}`

Item	Value
Content-Type	`application/x-www-form-urlencoded`

Body fields

Field	Type	Required	Description
`orderNo`	string	Yes	Your unique order reference (used for updates/deletes).
`clientBookId`	string	Yes	Your book reference ID (same ID used when content was registered in Kitaboo).
`userID`	string	Yes	Your unique user identifier (`userReferenceId` at launch).
`firstName`	string	Yes	User first name
`lastName`	string	Yes	User last name
`userName`	string	Yes	Login username (often email)
`password`	string	Yes	Password; if you use SSO only, a placeholder like `dummy` is acceptable.
`email`	string	Yes	Valid email (must contain `@` for mail-related jobs).
`endDate`	date	No	License end date `yyyy-MM-dd`. Omit for perpetual access.
`type`	int	No	`1` = individual; `5` = bulk (see your contract).

curl example

```
curl -sS -X POST "${REST_HOST}/ext/api/v3/order" \
  -H "Authorization: Bearer ${REST_ACCESS_TOKEN}" \
  -H "Content-Type: application/x-www-form-urlencoded" \
  --data-urlencode "orderNo=ORD-2026-001" \
  --data-urlencode "clientBookId=9780573800085" \
  --data-urlencode "endDate=2028-08-20" \
  --data-urlencode "userID=demo-user-005" \
  --data-urlencode "firstName=Demo" \
  --data-urlencode "lastName=User" \
  --data-urlencode "userName=demo005@example.com" \
  --data-urlencode "password=dummy" \
  --data-urlencode "email=demo005@example.com" \
  --data-urlencode "type=1"
```

Success response (example)

```
{
  "timestamp": "2026-03-24 15:39:19.19",
  "responseCode": "200",
  "responseMsg": "Order created successfully"
}
```

4.2 Update order — extend expiry or change keys

Item	Value
Method	`PUT`
URL	`{rest-host}/ext/api/v3/order`
Content-Type	`application/x-www-form-urlencoded`

Field	Type	Required	Description
`orderNo`	string	Yes	Same reference as create
`clientBookId`	string	Yes	Book reference
`endDate`	date	No	New end date `yyyy-MM-dd`; omit for perpetual
`keycount`	integer	No	License count; defaults depend on order `type`

curl example

```
curl -sS -X PUT "${REST_HOST}/ext/api/v3/order" \
  -H "Authorization: Bearer ${REST_ACCESS_TOKEN}" \
  -H "Content-Type: application/x-www-form-urlencoded" \
  --data-urlencode "orderNo=ORD-2026-001" \
  --data-urlencode "endDate=2028-08-20" \
  --data-urlencode "keycount=200" \
  --data-urlencode "clientBookId=9780573800085"
```

Success response (example)

```
{
  "timestamp": "2021-11-15 15:08:13.844",
  "responseCode": "200",
  "accesstoken": "1804925413967256",
  "responseMsg": "Order updated successfully"
}
```

4.3 Delete order

Item	Value
Method	`DELETE`
URL	`{rest-host}/ext/api/v3/order`

Field	Required	Description
`orderNo`	Yes	Order reference to remove

curl example

```
curl -sS -X DELETE "${REST_HOST}/ext/api/v3/order" \
  -H "Authorization: Bearer ${REST_ACCESS_TOKEN}" \
  -H "Content-Type: application/x-www-form-urlencoded" \
  --data-urlencode "orderNo=ORD-2026-001"
```

Success response (example)

```
{
  "responseCode": 200,
  "responseMsg": "Order deleted successfully"
}
```

---

5. Authentication — Microservices (Reader launch)

Reader launch uses the **API gateway** authorization service (`micro-authorization`), not the Rest `oauth2/authToken` endpoint.

Request

Item	Value
Method	`POST`
URL	`https://microservices.kitaboo.com/v1/micro-authorization/oauth/token?grant_type=client_credentials`
Authorization	`Basic {base64(client_id:client_secret)}`

Obtain `client_id` and `client_secret` from Kitaboo onboarding (integration credentials). **Do not commit** them to source control.

Example (curl)

```
GATEWAY="https://microservices.kitaboo.com"
# BASIC_TOKEN = Base64-encoded "client_id:client_secret" (integration credentials from Kitabo

curl -sS -X POST "${GATEWAY}/v1/micro-authorization/oauth/token?grant_type=client_credentials
  -H "Authorization: Basic ${BASIC_TOKEN}"
```

Sample response

```
{
  "access_token": "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9....",
  "token_type": "bearer",
  "expires_in": 899,
  "scope": "role_admin",
  "clientid": "1435",
  "schema": "client1435",
  "clientName": "Example Publishing",
  "userid": "322268",
  "region": "US"
}
```

Use **`access_token`** as `Authorization: Bearer {access_token}` on launch calls. Refresh before `expires_in` seconds elapse.

---

6. Launch bookshelf & book (`micro-lti`)

Service: `backend-micro-render` (exposed as `micro-lti` on the gateway)
Controller: `POST /auth/launch/object`
Full URL: `{gateway}/v1/micro-lti/auth/launch/object`

Headers

Header	Required	Description
`Authorization`	Yes	`Bearer {access_token}` from §5
`Content-Type`	Yes	`application/json`
`x-user-agent`	Recommended	Client browser/WebView user-agent string (used for device/store link selection)

Request body (JSON)

Field	Type	Required	Description
`userReferenceId`	string	Yes	Same value as Order API `userID` / your LMS user key
`objectReferenceId`	string	No	Same value as Order API `clientBookId`. **Omit** to open the **bookshelf** (library home). **Include** to open a **specific book**.

Optional `UserVO` fields exist for advanced flows; partner integrations typically only need the two reference IDs above.

6.1 Launch bookshelf (no title)

Opens the user's Kitaboo bookshelf (all entitled titles).

```
GATEWAY="https://microservices.kitaboo.com"
MS_ACCESS_TOKEN="<from micro-authorization>"

curl -sS -X POST "${GATEWAY}/v1/micro-lti/auth/launch/object" \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${MS_ACCESS_TOKEN}" \
  -H "x-user-agent: Mozilla/5.0 (Linux; Android 11) AppleWebKit/537.36 (KHTML, like Gecko) Ch
  -d '{
    "userReferenceId": "demo005@example.com"
  }'
```

Sample response

```
{
  "responseCode": 200,
  "responseMsg": "OK",
  "timestamp": "2026-09-22 06:53:40",
  "launchBookUrl": "https://read.kitaboo.com/reader/Launch.html?usertoken=...&storeLink=https
}
```

Client action: HTTP 302 redirect or open `launchBookUrl` in a browser/WebView.

6.2 Launch a specific book

```
curl -sS -X POST "${GATEWAY}/v1/micro-lti/auth/launch/object" \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${MS_ACCESS_TOKEN}" \
  -H "x-user-agent: Mozilla/5.0 (Linux; Android 11) AppleWebKit/537.36 (KHTML, like Gecko) Ch
  -d '{
    "userReferenceId": "demo005@example.com",
    "objectReferenceId": "23233"
  }'
```

Sample response

```
{
  "responseCode": 200,
  "responseMsg": "OK",
  "timestamp": "2026-09-22 07:24:30",
  "launchBookUrl": "https://read.kitaboo.com/reader/Launch.html?usertoken=...&bookID=30661&st
}
```

The response may include resolved internal `bookID` in the URL; your integration should still pass **`objectReferenceId`** (client reference), not the internal ID.

Launch failure response

When URL generation fails, `responseCode` is non-success and `launchBookUrl` may be absent:

```
{
  "responseCode": 500,
  "responseMsg": "Fail to generate URL",
  "timestamp": "2026-09-22 07:24:30"
}
```

Common causes: user has no entitlement (order missing), wrong `userReferenceId`, wrong `objectReferenceId`, or expired Bearer token.

---

7. HTTP response codes (Order API)

Code	Meaning
`100`	Empty required field
`101`	Invalid field
`102`	Invalid type
`106`	Duplicate data
`107`	Account inactive
`200`	Success (business `responseCode` in JSON body)
`400`	Bad request
`408`	Request timeout
`429`	Too many requests
`500`	Internal server error

---

8. Security & credentials

- Store **Rest OAuth** and **micro-authorization** credentials in a secret manager or environment variables.
- Never embed live `Authorization: Basic …` or JWT samples in git or public docs.
- Rotate credentials if they were ever pasted into tickets or chat.

- `launchBookUrl` contains a **user session token** — treat as sensitive; use HTTPS only; avoid logging full URLs in production.

Document control

Author	Issuer	Issue date
Vikas Shengale	Kitaboo	2026-09-22