Checkout Session

Author: Josh S. Sakweli, Backend Lead Team
Last Updated: 2026-05-23
Version: v2.0

Base URL: `https://apinexgate.glueauth.com/api/v1/`

Short Description: The Checkout Session API manages the complete checkout process for both e-commerce and event transactions. It handles session creation (with upfront wallet balance validation), inventory holds, payment processing via Wallet, Cash, and Free flows, group purchasing, and installment payments. Each session maintains state, inventory holds, and payment attempt tracking with automatic expiration handling.

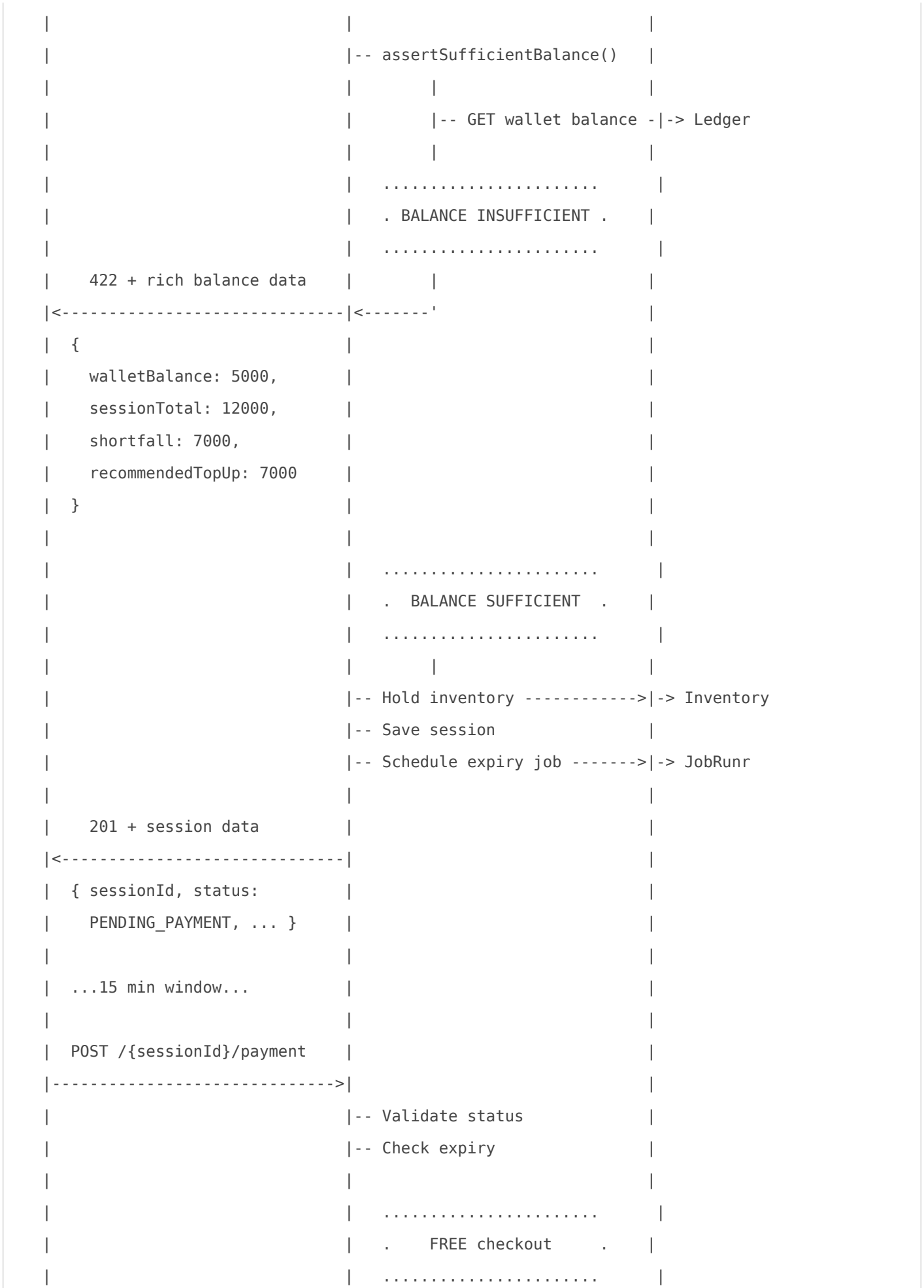
Hints:

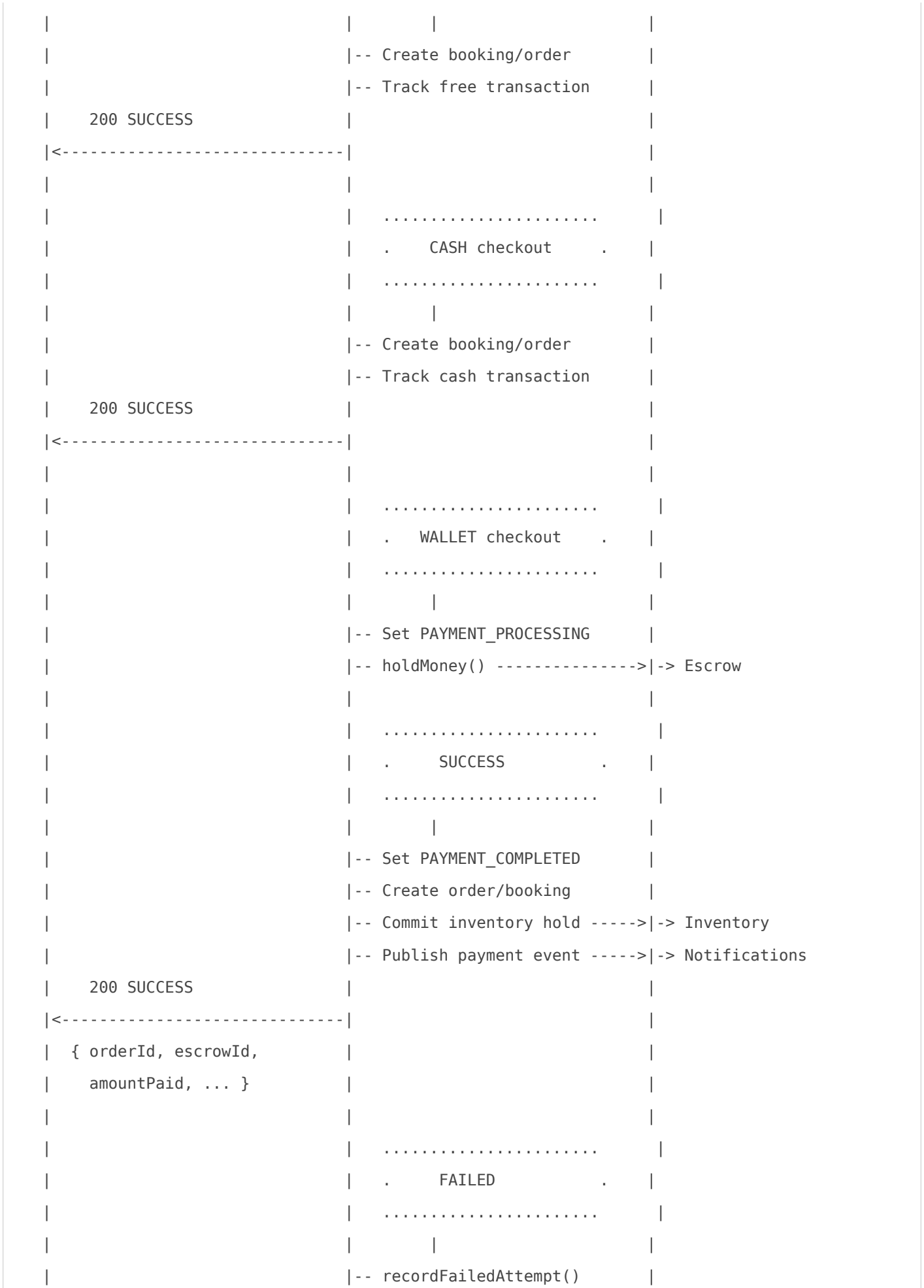
- **Wallet balance is validated at session creation time** — if insufficient, no session is created and a rich balance response is returned so the frontend can guide the user to top up
- All checkout sessions expire after 15 minutes by default
- Inventory is automatically held during active sessions and released upon expiration or cancellation
- Maximum 5 payment retry attempts allowed per session
- Group purchase sessions require WALLET payment method only
- Sessions in PAYMENT_PROCESSING status cannot be modified
- Installment payment only charges the down payment at checkout; monthly payments are handled by the scheduler
- All monetary values are in TZS (Tanzanian Shillings)
- Use ISO 8601 format for all datetime fields
- Sessions can only be updated when in PENDING_PAYMENT or PAYMENT_FAILED status

---

Full Checkout Flow Diagram

```
CLIENT                          BACKEND                          SYSTEMS
  |                               |                               |
  |  POST /checkout-sessions      |                               |
  |------------------------------>|                               |
  |                               |-- Validate request            |
  |                               |-- Calculate pricing           |
```


|-- assertSufficientBalance()
|-- GET wallet balance -|-> Ledger
422 + rich balance data
{
walletBalance: 5000,
sessionTotal: 12000,
shortfall: 7000,
recommendedTopUp: 7000
}
BALANCE SUFFICIENT
|-- Hold inventory ------------>|-> Inventory
|-- Save session
|-- Schedule expiry job ------->|-> JobRunr
201 + session data
{ sessionId, status:
PENDING_PAYMENT, ... }
...15 min window...
POST /{sessionId}/payment
|-- Validate status
|-- Check expiry
FREE checkout


Create booking/order
Track free transaction
200 SUCCESS
CASH checkout
Create booking/order
Track cash transaction
200 SUCCESS
WALLET checkout
Set PAYMENT_PROCESSING
holdMoney()
Escrow
SUCCESS
Set PAYMENT_COMPLETED
Create order/booking
Commit inventory hold
Inventory
Publish payment event
Notifications
200 SUCCESS
{ orderId, escrowId,
amountPaid, ... }
FAILED
recordFailedAttempt()

```
  |                              |-- Release reservation         |
  |                              |   (events only)               |
  |    200 FAILED                |                               |
  |<-----------------------------|                               |
  |  { success: false,           |                               |
  |    canRetry: true/false }    |                               |
  |                              |                               |
  |  POST /{sessionId}/retry     |                               |
  |----------------------------->|                               |
  |                              |-- Check status = FAILED       |
  |                              |-- Check attempts < 5          |
  |                              |-- Re-validate inventory       |
  |                              |-- Check wallet balance        |
  |                              |-- Re-hold inventory           |
  |                              |-- Reset to PENDING_PAYMENT    |
  |                              |-- processPayment() ...        |
  |                              |   (same wallet flow above)    |
```

Session Type Routing

```
POST /checkout-sessions
  sessionType?
  .
  ├─ REGULAR_DIRECTLY ──> validate 1 item ──> balance check ──> hold inventory ──> session
  |
  ├─ REGULAR_CART ──────> validate cart ────> balance check ──> hold inventory ──> session
  |
  ├─ GROUP_PURCHASE ────> validate group ──> balance check ──> session (no hold, group-level)
  |
  └─ INSTALLMENT ───────> calculate down payment ──> balance check ──> hold inventory ──>
session


POST /e-events/checkout
  ticketPricingType?
  .
  ├─ PAID ──> balance check ──> create payment intent ──> reserve ticket ──> session
  |
```

```
   └─ FREE ──> (no balance check) ──> reserve ticket ──> session
```

Standard Response Format

All API responses follow a consistent structure using our Globe Response Builder pattern:

Success Response Structure

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Operation completed successfully",
  "action_time": "2025-10-02T10:30:45",
  "data": {}
}
```

Error Response Structure

```
{
  "success": false,
  "httpStatus": "BAD_REQUEST",
  "message": "Error description",
  "action_time": "2025-10-02T10:30:45",
  "data": "Error description"
}
```

Insufficient Balance Error Structure (422)

This is a special structured error returned when wallet balance is insufficient **at session creation time**. The `data` field contains rich balance details so the frontend can guide the user to top up.

```
{
  "success": false,
  "httpStatus": "UNPROCESSABLE_ENTITY",
  "message": "Insufficient wallet balance to complete checkout",
  "action_time": "2025-10-02T10:30:45",
  "data": {
```

```
    "walletBalance": 5000.00,
    "sessionTotal": 12000.00,
    "shortfall": 7000.00,
    "hasSufficientBalance": false,
    "recommendedTopUp": 7000.00,
    "pspMinimum": 500.00,
    "currency": "TZS"
  }
}
```

Field	Description
`walletBalance`	Current wallet balance of the user
`sessionTotal`	Total amount required for this checkout
`shortfall`	How much is missing (`sessionTotal - walletBalance`)
`hasSufficientBalance`	Always `false` when this error is returned
`recommendedTopUp`	Suggested top-up amount (at least `shortfall`, rounded up to PSP minimum)
`pspMinimum`	Minimum top-up amount accepted by the payment provider
`currency`	Always `TZS`

Standard Response Fields

Field	Type	Description
`success`	boolean	Always `true` for successful operations, `false` for errors
`httpStatus`	string	HTTP status name (OK, CREATED, BAD_REQUEST, NOT_FOUND, etc.)
`message`	string	Human-readable message describing the operation result
`action_time`	string	ISO 8601 timestamp of when the response was generated
`data`	object/string	Response payload for success, error details for failures

---

HTTP Method Badge Standards

For better visual clarity, all endpoints use colored badges for HTTP methods with the following standard colors:

- **GET** - `GET` - Green (Safe, read-only operations)
- **POST** - `POST` - Blue (Create new resources)
- **PATCH** - `PATCH` - Orange (Partial updates)
- **DELETE** - `DELETE` - Red (Remove resources)

Endpoints

1. Create Checkout Session

Purpose: Creates a new checkout session for processing a purchase. Wallet balance is validated upfront — if insufficient, no session is created and the caller receives rich balance data to guide the user. Supports multiple checkout types: direct product purchase, cart checkout, group purchasing, and installment payments.

Endpoint: `POST` `{base_url}/checkout-sessions`

Access Level: □□ Protected (Requires Authentication)

Authentication: Bearer Token required in Authorization header

Request Headers:

Header	Type	Required	Description
Authorization	string	Yes	Bearer token for authenticated user
Content-Type	string	Yes	Must be `application/json`

Request JSON Sample:

```
{
  "sessionType": "REGULAR_DIRECTLY",
  "items": [
    {
      "productId": "a1b2c3d4-e5f6-7890-abcd-ef1234567890",
      "quantity": 2
    }
  ],
```

```
  "shippingAddressId": "f1e2d3c4-b5a6-7890-cdef-123456789abc",
  "shippingMethodId": "standard-shipping",
  "metadata": {
    "couponCode": "SAVE20",
    "referralCode": "REF123",
    "notes": "Please handle with care"
  },
  "installmentPlanId": null,
  "groupInstanceId": null,
  "downPaymentPercent": 20
}
```

Request Body Parameters:

Parameter	Type	Required	Description	Validation
sessionType	string	Yes	Type of checkout session	enum: REGULAR_DIRECTLY, REGULAR_CART, GROUP_PURCHASE, INSTALLMENT
items	array	Conditional	Array of items to checkout. Required for REGULAR_DIRECTLY and GROUP_PURCHASE	For REGULAR_DIRECTLY: exactly 1 item. For GROUP_PURCHASE: exactly 1 item. Not used for REGULAR_CART
items[].productId	string (UUID)	Yes (if items provided)	Product identifier	Valid UUID format
items[].quantity	integer	Yes (if items provided)	Quantity to purchase	Min: 1, must not exceed product constraints
shippingAddressId	string (UUID)	Yes	User's shipping address identifier	Valid UUID format, must belong to authenticated user
shippingMethodId	string	Yes	Selected shipping method identifier	Must be valid shipping method
metadata	object	No	Additional metadata for the session	Key-value pairs for coupons, referrals, notes, etc.
installmentPlanId	string (UUID)	Conditional	Installment plan identifier (INSTALLMENT type only)	Valid UUID format

Parameter	Type	Required	Description	Validation
downPaymentPercent	integer	Conditional	Down payment percentage (INSTALLMENT type only)	Must satisfy plan's min/max constraints
groupInstanceId	string (UUID)	No	Group instance to join (GROUP_PURCHASE type only)	Valid UUID format, group must be joinable. Null = create new group
groupName	string	Conditional	Name for new group (GROUP_PURCHASE, groupInstanceId null)	Must be unique per product

Success Response JSON Sample:


```
{
  "success": true,
  "httpStatus": "CREATED",
  "message": "Checkout session created successfully",
  "action_time": "2025-10-02T14:30:45",
  "data": {
    "sessionId": "c1d2e3f4-a5b6-7890-cdef-123456789abc",
    "sessionType": "REGULAR_DIRECTLY",
    "status": "PENDING_PAYMENT",
    "customerId": "u1s2e3r4-i5d6-7890-abcd-ef1234567890",
    "customerUserName": "john_doe",
    "items": [
      {
        "productId": "a1b2c3d4-e5f6-7890-abcd-ef1234567890",
        "productName": "Premium Wireless Headphones",
        "productSlug": "premium-wireless-headphones",
        "productPrimaryMedia": {
          "fileId": "8f2a1c3d-1234-4a5b-9c6d-abcdef012345",
          "mediaType": "IMAGE",
          "status": "READY",
          "thumbUrl": "https://cdn.nextgate.com/products/headphones-001/thumb.webp",
          "mimeType": "image/webp"
        },
        "quantity": 2,
        "unitPrice": 150000.00,
        "subtotal": 300000.00,
        "tax": 0.00,
```

```
    "total": 300000.00,
    "shopId": "s1h2o3p4-i5d6-7890-abcd-ef1234567890",
    "shopName": "TechWorld Electronics",
    "availableForCheckout": true,
    "availableQuantity": 50
  }
],
"pricing": {
  "subtotal": 300000.00,
  "shippingCost": 5000.00,
  "tax": 0.00,
  "total": 305000.00,
  "currency": "TZS"
},
"shippingAddress": {
  "fullName": "John Doe",
  "addressLine1": "123 Main Street",
  "addressLine2": "Apartment 4B",
  "city": "Dar es Salaam",
  "state": "Dar es Salaam Region",
  "postalCode": "12345",
  "country": "Tanzania",
  "phone": "+255123456789"
},
"shippingMethod": {
  "id": "standard-shipping",
  "name": "Standard Shipping",
  "carrier": "DHL",
  "cost": 5000.00,
  "estimatedDays": "3-5 business days",
  "estimatedDelivery": "2025-10-07T14:30:45"
},
"paymentAttempts": [],
"inventoryHeld": true,
"inventoryHoldExpiresAt": "2025-10-02T14:45:45",
"metadata": {
  "couponCode": "SAVE20",
  "referralCode": "REF123",
  "notes": "Please handle with care"
},
```

```
    "expiresAt": "2025-10-02T14:45:45",
    "createdAt": "2025-10-02T14:30:45",
    "updatedAt": "2025-10-02T14:30:45",
    "completedAt": null,
    "createdOrderIds": [],
    "cartId": null
  }
}
```

Success Response Fields:

Field	Description
sessionId	Unique identifier for the checkout session
sessionType	Type of checkout (REGULAR_DIRECTLY, REGULAR_CART, GROUP_PURCHASE, INSTALLMENT)
status	Current status of the session (always PENDING_PAYMENT on creation)
customerId	User ID who created the session
customerUserName	Username of the customer
items	Array of checkout items with product details, pricing, and availability. Each item's image is `productPrimaryMedia` (`{fileId, mediaType, status, thumbUrl, mimeType}`) — the lean, resolved primary-media shape, same as cart/wishlist/marketplace
pricing	Summary of all pricing calculations
pricing.total	Final amount to be charged (what was validated against wallet balance)
pricing.currency	Always TZS
shippingAddress	Complete shipping address details
shippingMethod	Selected shipping method details
paymentIntent	Payment processing details. Field is declared but not currently populated by any checkout flow — omitted (null) in real responses today
paymentAttempts	Empty array on creation
inventoryHeld	Whether inventory is currently held (false for GROUP_PURCHASE)
inventoryHoldExpiresAt	When the inventory hold will be released
expiresAt	When this checkout session expires (15 minutes from creation)

Field	Description
createdOrderIds	Empty array until payment succeeds; populated with the created order ID(s) after
cartId	Cart ID if session was created from cart, null otherwise

Error Responses:

Insufficient Wallet Balance (422) — includes rich top-up data:

```
{
  "success": false,
  "httpStatus": "UNPROCESSABLE_ENTITY",
  "message": "Insufficient wallet balance to complete checkout",
  "action_time": "2025-10-02T14:30:45",
  "data": {
    "walletBalance": 150000.00,
    "sessionTotal": 285000.00,
    "shortfall": 135000.00,
    "hasSufficientBalance": false,
    "recommendedTopUp": 135000.00,
    "pspMinimum": 500.00,
    "currency": "TZS"
  }
}
```

Bad Request - Invalid Session Type (400):

```
{
  "success": false,
  "httpStatus": "BAD_REQUEST",
  "message": "REGULAR_DIRECTLY checkout supports only 1 item. Use REGULAR_CART for multiple
items.",
  "action_time": "2025-10-02T14:30:45",
  "data": "REGULAR_DIRECTLY checkout supports only 1 item. Use REGULAR_CART for multiple
items."
}
```

Bad Request - Insufficient Inventory (400):

```
{
  "success": false,
```

```
  "httpStatus": "BAD_REQUEST",
  "message": "Insufficient stock. Available: 3, Requested: 5",
  "action_time": "2025-10-02T14:30:45",
  "data": "Insufficient stock. Available: 3, Requested: 5"
}
```

Validation Error (422):

```
{
  "success": false,
  "httpStatus": "UNPROCESSABLE_ENTITY",
  "message": "Validation failed",
  "action_time": "2025-10-02T14:30:45",
  "data": {
    "sessionType": "must not be null",
    "items[0].quantity": "must be greater than or equal to 1",
    "shippingAddressId": "must not be null"
  }
}
```

Not Found - Product Not Found (404):

```
{
  "success": false,
  "httpStatus": "NOT_FOUND",
  "message": "Product not found",
  "action_time": "2025-10-02T14:30:45",
  "data": "Product not found"
}
```

Unauthorized (401):

```
{
  "success": false,
  "httpStatus": "UNAUTHORIZED",
  "message": "Authentication token is required",
  "action_time": "2025-10-02T14:30:45",
  "data": "Authentication token is required"
}
```

---

2. Get Checkout Session by ID

Purpose: Retrieves detailed information about a specific checkout session by its ID. Only the session owner can access their session.

Endpoint: `GET` `{base_url}/checkout-sessions/{sessionId}`

Access Level: □□ Protected (Requires Authentication and Ownership)

Authentication: Bearer Token required in Authorization header

Path Parameters:

Parameter	Type	Required	Description	Validation
sessionId	string (UUID)	Yes	Unique identifier of the checkout session	Valid UUID format

Success Response JSON Sample:


```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Checkout session retrieved successfully",
  "action_time": "2025-10-02T14:35:45",
  "data": {
    "sessionId": "c1d2e3f4-a5b6-7890-cdef-123456789abc",
    "sessionType": "REGULAR_DIRECTLY",
    "status": "PENDING_PAYMENT",
    "customerId": "u1s2e3r4-i5d6-7890-abcd-ef1234567890",
    "customerUserName": "john_doe",
    "items": [],
    "pricing": {},
    "shippingAddress": {},
    "shippingMethod": {},
    "paymentIntent": {},
    "paymentAttempts": [],
    "inventoryHeld": true,
    "inventoryHoldExpiresAt": "2025-10-02T14:45:45",
    "metadata": { "couponCode": "SAVE20" },
    "expiresAt": "2025-10-02T14:45:45",
    "createdAt": "2025-10-02T14:30:45",
```

```
    "updatedAt": "2025-10-02T14:30:45",
    "completedAt": null,
    "createdOrderId": null,
    "cartId": null
  }
}
```

Error Responses:

Not Found - Session Not Found or No Permission (404):

```
{
  "success": false,
  "httpStatus": "NOT_FOUND",
  "message": "Checkout session not found or you don't have permission to access it",
  "action_time": "2025-10-02T14:35:45",
  "data": "Checkout session not found or you don't have permission to access it"
}
```

---

3. Get My Checkout Sessions

Purpose: Retrieves all checkout sessions belonging to the authenticated user, ordered by creation date (newest first).

Endpoint: `GET` `{base_url}/checkout-sessions`

Access Level: □□ Protected (Requires Authentication)

Success Response Fields:

Field	Description
sessionId	Unique identifier for the checkout session
sessionType	Type of checkout session
status	Current status of the session
itemCount	Number of items in the checkout
totalAmount	Total amount to be paid in TZS
currency	Currency code (TZS)
expiresAt	When this session expires

Field	Description
createdAt	When this session was created
isExpired	Whether the session has expired
canRetryPayment	true only if status is PAYMENT_FAILED and not expired and attempts < 5
itemPreviews	Array of preview information for items

---

4. Get My Active Checkout Sessions

Purpose: Retrieves only active checkout sessions (PENDING_PAYMENT or PAYMENT_FAILED status) that haven't expired yet, ordered by creation date (newest first).

Endpoint: `GET` `{base_url}/checkout-sessions/active`

Access Level: □□ Protected (Requires Authentication)

Notes: Response structure is the same as Get My Checkout Sessions but filtered to active sessions only. `isExpired` is always `false` in this response.

---

5. Update Checkout Session

Purpose: Updates an existing checkout session. Can modify shipping address, shipping method, or metadata. Only sessions in PENDING_PAYMENT or PAYMENT_FAILED status can be updated.

Endpoint: `PATCH` `{base_url}/checkout-sessions/{sessionId}`

Access Level: □□ Protected (Requires Authentication and Ownership)

Path Parameters:

Parameter	Type	Required	Description
sessionId	string (UUID)	Yes	Unique identifier of the checkout session

Request Body Parameters:

Parameter	Type	Required	Description
shippingAddressId	string (UUID)	No	New shipping address identifier

Parameter	Type	Required	Description
shippingMethodId	string	No	New shipping method (triggers pricing recalculation)
metadata	object	No	Key-value pairs, merged with existing metadata

Error Responses:

Cannot Update Completed Session (400):

```
{
  "success": false,
  "httpStatus": "BAD_REQUEST",
  "message": "Cannot update a completed checkout session",
  "action_time": "2025-10-02T14:50:45",
  "data": "Cannot update a completed checkout session"
}
```

---

6. Cancel Checkout Session

Purpose: Cancels an existing checkout session and releases held inventory. Cannot cancel sessions that are completed or have successful payment.

Endpoint: `DELETE` `{base_url}/checkout-sessions/{sessionId}/cancel`

Access Level: □□ Protected (Requires Authentication and Ownership)

Success Response:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Checkout session cancelled successfully",
  "action_time": "2025-10-02T14:55:45",
  "data": null
}
```

Error Responses:

Cannot Cancel Completed (400):

```
{
  "success": false,
  "httpStatus": "BAD_REQUEST",
  "message": "Cannot cancel - payment has been completed. Please contact support.",
  "action_time": "2025-10-02T14:55:45",
  "data": "Cannot cancel - payment has been completed. Please contact support."
}
```


Already Cancelled (400):


```
{
  "success": false,
  "httpStatus": "BAD_REQUEST",
  "message": "Checkout session is already cancelled",
  "action_time": "2025-10-02T14:55:45",
  "data": "Checkout session is already cancelled"
}
```


---

7. Process Payment

Purpose: Initiates payment processing for a checkout session in PENDING_PAYMENT status. Routes to the appropriate payment processor (WALLET, CASH, FREE) based on the session amount and payment method.

Endpoint: `POST` `{base_url}/checkout-sessions/{sessionId}/process-payment`

Access Level: □□ Protected (Requires Authentication and Ownership)

Path Parameters:

Parameter	Type	Required	Description
sessionId	string (UUID)	Yes	Unique identifier of the checkout session

Success Response JSON Sample:


```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Payment completed successfully. Your order is being processed.",
```

```
  "action_time": "2025-10-02T15:00:45",
  "data": {
    "success": true,
    "status": "SUCCESS",
    "message": "Payment completed successfully. Your order is being processed.",
    "checkoutSessionId": "c1d2e3f4-a5b6-7890-cdef-123456789abc",
    "escrowId": "esc-uuid",
    "escrowNumber": "ESC-20251002-001",
    "orderId": "ord-uuid",
    "paymentMethod": "WALLET",
    "amountPaid": 285000.00,
    "platformFee": 5700.00,
    "sellerAmount": 279300.00,
    "currency": "TZS"
  }
}
```

Success Response Fields:

Field	Description
success	Whether the payment was successful
status	SUCCESS, PENDING, or FAILED
checkoutSessionId	The session that was paid
escrowId	Escrow account holding the funds
escrowNumber	Human-readable escrow reference
orderId	Order or booking ID created after payment
paymentMethod	WALLET, CASH, or FREE
amountPaid	Total amount charged in TZS
platformFee	Platform fee deducted
sellerAmount	Amount the seller receives
currency	Always TZS

Error Responses:

Session Not in PENDING_PAYMENT status (400):

```
{
  "success": false,
```

```
  "httpStatus": "BAD_REQUEST",
  "message": "Cannot process payment - session is not pending: PAYMENT_COMPLETED",
  "action_time": "2025-10-02T15:00:45",
  "data": "Cannot process payment - session is not pending: PAYMENT_COMPLETED"
}
```


Session Expired (400):


```
{
  "success": false,
  "httpStatus": "BAD_REQUEST",
  "message": "Checkout session has expired",
  "action_time": "2025-10-02T15:00:45",
  "data": "Checkout session has expired"
}
```


---

8. Retry Payment

Purpose: Retries payment for a session in PAYMENT_FAILED status. Re-validates inventory availability, checks wallet balance, re-holds inventory, and extends session expiration before retrying. Maximum 5 total attempts.

Endpoint: `POST` `{base_url}/checkout-sessions/{sessionId}/retry-payment`

Access Level: □□ Protected (Requires Authentication and Ownership)

Error Responses:

Invalid Status (400):


```
{
  "success": false,
  "httpStatus": "BAD_REQUEST",
  "message": "Cannot retry payment - session status: PENDING_PAYMENT. Expected:
PAYMENT_FAILED",
  "action_time": "2025-10-02T15:10:45",
  "data": "Cannot retry payment - session status: PENDING_PAYMENT. Expected: PAYMENT_FAILED"
}
```


Max Attempts Exceeded (400):

```
{
  "success": false,
  "httpStatus": "BAD_REQUEST",
  "message": "Maximum payment attempts (5) exceeded. Please create a new checkout session.",
  "action_time": "2025-10-02T15:10:45",
  "data": "Maximum payment attempts (5) exceeded. Please create a new checkout session."
}
```


Insufficient Wallet Balance on Retry (400):


```
{
  "success": false,
  "httpStatus": "BAD_REQUEST",
  "message": "Insufficient wallet balance. Required: 285000 TZS, Available: 150000 TZS. Please top up your wallet.",
  "action_time": "2025-10-02T15:10:45",
  "data": "Insufficient wallet balance. Required: 285000 TZS, Available: 150000 TZS. Please top up your wallet."
}
```


Product No Longer Available (400):


```
{
  "success": false,
  "httpStatus": "BAD_REQUEST",
  "message": "Product 'Premium Wireless Headphones' is no longer available in requested quantity. Please create a new checkout session.",
  "action_time": "2025-10-02T15:10:45",
  "data": "Product 'Premium Wireless Headphones' is no longer available in requested quantity. Please create a new checkout session."
}
```

Checkout Session Types

REGULAR_DIRECTLY

Direct product purchase without using a cart. Must include exactly 1 item.

- Best for "Buy Now" buttons
- Balance is checked against `pricing.total` (includes shipping)
- Inventory is held immediately on session creation

REGULAR_CART

Checkout from existing shopping cart. Items are fetched automatically from the user's active cart.

- Multi-item purchase flow
- Items array not required in request
- Balance is checked against `pricing.total` (includes shipping)
- Inventory is held for all cart items

GROUP_PURCHASE

Group buying checkout where multiple users purchase the same product at a discounted group price.

- Single item at `product.groupPrice`
- Balance is checked against `groupPrice × quantity`
- WALLET payment method only
- Can join existing group (provide `groupInstanceId`) or create new group (provide `groupName`)
- No session-level inventory hold — inventory is held at the group level after payment

Create New Group:

```
{
  "sessionType": "GROUP_PURCHASE",
  "items": [{ "productId": "prod-uuid", "quantity": 2 }],
  "shippingAddressId": "addr-uuid",
  "shippingMethodId": "standard",
  "groupName": "My Winning Group"
}
```

Join Existing Group:

```
{
  "sessionType": "GROUP_PURCHASE",
  "items": [{ "productId": "prod-uuid", "quantity": 3 }],
  "shippingAddressId": "addr-uuid",
  "shippingMethodId": "standard",
```

```
  "groupInstanceId": "group-uuid-to-join"
}
```

INSTALLMENT

Split payment over multiple months. Only the down payment is charged at checkout; remaining monthly payments are processed automatically by the scheduler.

- Balance is checked against `downPaymentAmount` only (not the full product price)
- `installmentPlanId` and `downPaymentPercent` are required
- `pricing.total` in the session response reflects the down payment only
- Inventory is held immediately on session creation

```
{
  "sessionType": "INSTALLMENT",
  "items": [{ "productId": "prod-uuid", "quantity": 1 }],
  "shippingAddressId": "addr-uuid",
  "shippingMethodId": "standard",
  "installmentPlanId": "plan-uuid",
  "downPaymentPercent": 20
}
```

Checkout Session Status Flow

Status Definitions

Status	Description	Can Update?	Can Cancel?	Can Pay?	Can Retry?
PENDING_PAYMENT	Session created, awaiting payment	Yes	Yes	Yes	No
PAYMENT_PROCESSING	Payment in progress	No	No	No	No
PAYMENT_FAILED	Payment failed, can retry	Yes	Yes	No	Yes
PAYMENT_COMPLETED	Payment successful	No	No	No	No
EXPIRED	Session expired	No	No	No	No
CANCELLED	User cancelled	No	No	No	No

Status	Description	Can Update?	Can Cancel?	Can Pay?	Can Retry?
COMPLETED	Free/cash order completed	No	No	No	No

Status Transition Flow

```
[Session Creation]
        |
        | balance check passes
        v
PENDING_PAYMENT
    |
    |-- user cancels ──────────────────> CANCELLED
    |                                    (inventory released)
    |
    |-- 15 min timeout ────────────────> EXPIRED
    |                                    (inventory released)
    |
    |-- processPayment()
        |
        |-- amount = 0 ────────────────> COMPLETED (free)
        |
        |-- CASH ──────────────────────> COMPLETED (cash)
        |
        |-- WALLET ────────────────────> PAYMENT_PROCESSING
                                            |
                                            |-- success ──> PAYMENT_COMPLETED
                                            |                (order/booking created,
                                            |                 inventory committed)
                                            |
                                            |-- failure ──> PAYMENT_FAILED
                                                           (inventory released for
events,
                                                            max 5 retries)
                                                           |
                                                           |-- retryPayment() ──>
PENDING_PAYMENT
                                                           |
(inventory re-held)
```

```
                                                    |
                                                    |-- 5 attempts ————> EXPIRED
```

Wallet Balance Check Endpoint

For cases where the frontend wants to proactively check balance against an existing session (e.g., before showing the "Pay Now" button), use:

Endpoint: `GET` `{base_url}/wallet/checkout-balance-check?sessionId={id}&domain={PRODUCT|EVENT}`

Success Response:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Checkout balance check completed",
  "data": {
    "walletBalance": 150000.00,
    "sessionTotal": 285000.00,
    "shortfall": 135000.00,
    "hasSufficientBalance": false,
    "recommendedTopUp": 135000.00,
    "pspMinimum": 500.00,
    "currency": "TZS"
  }
}
```

Note: This endpoint always returns 200 — it never throws. Use `hasSufficientBalance` to determine if the user can pay. This is the "soft check" for an existing session; the "hard check" (which blocks session creation) happens automatically inside `POST /checkout-sessions`.

Payment Methods Supported

WALLET (Default)

Internal wallet system. Default if no payment method is specified.

- Balance validated at **session creation** (hard block) and again at **payment time** (safety net)
- Instant processing via escrow
- Funds held in escrow until order is confirmed/delivered

CASH

Pay in cash on delivery or at the point of event check-in.

- No pre-payment required
- Order/booking created immediately
- Applicable to both product and event checkouts

FREE

Zero-amount checkout (free products or free event tickets).

- Handled automatically when `pricing.total = 0`
- No payment method required
- Order/booking created immediately

CREDIT_CARD / MOBILE_MONEY

Status: Planned — not yet implemented.

Inventory Management

Hold Mechanism

Session Type	Hold Created At	Hold Released At
REGULAR_DIRECTLY	Session creation	Expiry, cancellation, or payment success (committed)
REGULAR_CART	Session creation	Expiry, cancellation, or payment success (committed)
GROUP_PURCHASE	After payment, at group level	Group expiry or group failure
INSTALLMENT	Session creation	Expiry, cancellation, or payment success (committed)

On successful payment, holds are **committed** (stock permanently deducted). On failure/expiry/cancellation, holds are **released** (stock returned to available).

Session Expiration

Default: 15 minutes from creation.

Extended when: Payment retry is initiated (adds 15 minutes).

On expiry:

1. Status → EXPIRED
2. Held inventory released
3. Session cannot be updated, paid, or cancelled
4. User must create a new checkout session

Payment Attempts Tracking

Maximum 5 attempts per session. Each attempt records:

- Attempt number (1–5)
- Payment method used
- Status: SUCCESS, FAILED, or RETRY_INITIATED
- Error message (if failed)
- Timestamp and transaction ID

After 5 failed attempts, session status moves to EXPIRED and inventory is released.

Error Handling Best Practices

Frontend Checklist

Before calling POST /checkout-sessions:

- Ensure the user has a shipping address saved
- No need to pre-check balance — the API returns rich balance data if insufficient

On 422 Insufficient Balance response:

- Read `data.shortfall` to show how much the user is short
- Read `data.recommendedTopUp` to pre-fill a top-up amount
- Navigate the user to the wallet top-up screen
- Once topped up, retry `POST /checkout-sessions` — do not store the failed session

During active session (PENDING_PAYMENT):

- Show a countdown timer using `expiresAt`
- On expiry, prompt user to create a new session

On payment failure:

- Show `canRetryPayment` to decide whether to show a retry button
- Show remaining attempts (`5 - paymentAttemptCount`)
- On retry, call `POST /{sessionId}/retry-payment` — no need to create a new session

Integration Examples

Example 1: Direct Product Purchase

```
1. POST /checkout-sessions
   → 422 if balance insufficient (show top-up screen with data.recommendedTopUp)
   → 201 with sessionId if balance OK

2. POST /checkout-sessions/{sessionId}/process-payment
   → 200 with orderId on success
```

Example 2: Cart Checkout

```
1. POST /checkout-sessions  { sessionType: REGULAR_CART, ... }
   → 422 or 201

2. (optional) PATCH /checkout-sessions/{sessionId}  { shippingMethodId: "express" }

3. POST /checkout-sessions/{sessionId}/process-payment
```

Example 3: Group Purchase

```
1. POST /checkout-sessions  { sessionType: GROUP_PURCHASE, groupName: "...", ... }
   → 422 if balance < groupPrice × qty
   → 201 with sessionId

2. POST /checkout-sessions/{sessionId}/process-payment
   (WALLET only)
```

Example 4: Installment

```
1. POST /checkout-sessions  { sessionType: INSTALLMENT, installmentPlanId: "...",
downPaymentPercent: 20 }
   → 422 if balance < downPaymentAmount
   → 201 with sessionId (pricing.total = down payment only)

2. POST /checkout-sessions/{sessionId}/process-payment
   → charges down payment only; monthly payments handled by scheduler
```

Example 5: Payment Retry

```
1. GET /checkout-sessions/active
   → find session with status: PAYMENT_FAILED, canRetryPayment: true

2. POST /checkout-sessions/{sessionId}/retry-payment
   → re-validates inventory + balance
   → re-holds inventory
   → processes payment
```

Rate Limiting

Endpoint	Limit
Create Checkout	20 req/min per user
Get Sessions	60 req/min per user
Process / Retry Payment	10 req/min per user
Update / Cancel	30 req/min per user

Revision #6
Created 2 October 2025 06:00:25 by Admin Qbit
Updated 9 July 2026 08:03:27 by Admin Qbit