Order Management

Author: Josh S. Sakweli, Backend Lead Team **Last Updated:** 2026-06-10 **Version:** v1.0

Base URL: `api/v1/e-commerce/orders`

Short Description: The Order Management API handles the complete order lifecycle for the NextGate e-commerce platform. It supports multiple purchase types, order tracking, shipping management, delivery confirmation with 6-digit codes, escrow integration, and digital product downloads.

Hints:

- All endpoints require Bearer token authentication
- Order sources: DIRECT_PURCHASE, CART_PURCHASE, DIGITAL_PURCHASE, INSTALLMENT, GROUP_PURCHASE
- Delivery confirmation uses a 6-digit code (SHA-256 hashed with salt, expires in 30 days, max 5 attempts)
- Digital orders have `deliveryStatus: NOT_APPLICABLE`
- Confirm-delivery response is returned directly (not wrapped in the standard response envelope)
- Every order detail response includes a `timeline` array — ordered list of status steps with timestamps. Steps not yet reached have `timestamp: null` and `isCompleted: false`

Endpoints

1. Get Order by ID

Purpose: Retrieve detailed information about a specific order.

Endpoint: `GET` `{base}/{orderId}`

Access Level: □□ Protected (Buyer or Seller only)

Authentication: Bearer Token

Request Headers:

Header	Type	Required	Description

Authorization	string	Yes	Bearer token for authentication

Path Parameters:

Parameter	Type	Required	Description
orderId	UUID	Yes	Unique identifier of the order

Response JSON Sample:


```
{
  "success": true,
  "message": "Order retrieved successfully",
  "data": {
    "orderId": "550e8400-e29b-41d4-a716-446655440000",
    "orderNumber": "ORD-2025-12345",
    "buyer": {
      "accountId": "123e4567-e89b-12d3-a456-426614174000",
      "userName": "johndoe",
      "email": "john@example.com",
      "firstName": "John",
      "lastName": "Doe"
    },
    "seller": {
      "shopId": "789e0123-e45b-67d8-a901-234567890abc",
      "shopName": "TechStore",
      "shopLogo": "https://cdn.example.com/shops/techstore.png",
      "shopSlug": "techstore"
    },
    "productOrderStatus": "SHIPPED",
    "deliveryStatus": "IN_TRANSIT",
    "productOrderSource": "DIRECT_PURCHASE",
    "items": [
      {
        "orderItemId": "111e2222-e33b-44d5-a666-777788889999",
        "productId": "abc12345-def6-7890-ghij-klmnopqrstuv",
        "productName": "Wireless Headphones",
        "productSlug": "wireless-headphones",
        "productImage": "https://cdn.example.com/products/headphones.jpg",
        "productType": "PHYSICAL",
```

```
    "fileIds": null,
    "quantity": 2,
    "unitPrice": 85000.00,
    "subtotal": 170000.00,
    "tax": 0.00,
    "total": 170000.00
  }
],
"subtotal": 170000.00,
"shippingFee": 5000.00,
"tax": 0.00,
"totalAmount": 175000.00,
"platformFee": 8750.00,
"sellerAmount": 166250.00,
"currency": "TZS",
"paymentMethod": "MPESA",
"amountPaid": 175000.00,
"amountRemaining": 0.00,
"deliveryAddress": "123 Main St, Dar es Salaam, Tanzania",
"trackingNumber": "TRACK-550E8400",
"carrier": "NextGate Shipping",
"isDeliveryConfirmed": false,
"deliveryConfirmedAt": null,
"orderedAt": "2025-10-20T14:30:00",
"shippedAt": "2025-10-21T09:15:00",
"deliveredAt": null,
"cancelledAt": null,
"cancellationReason": null,
"timeline": [
  {
    "status": "ORDER_PLACED",
    "label": "Order Placed",
    "timestamp": "2025-10-20T14:30:00",
    "isCompleted": true,
    "note": null
  },
  {
    "status": "SHIPPED",
    "label": "Shipped",
    "timestamp": "2025-10-21T09:15:00",
```

```
        "isCompleted": true,
        "note": "NextGate Shipping · TRACK-550E8400"
      },
      {
        "status": "DELIVERED",
        "label": "Delivered",
        "timestamp": null,
        "isCompleted": false,
        "note": null
      },
      {
        "status": "COMPLETED",
        "label": "Order Completed",
        "timestamp": null,
        "isCompleted": false,
        "note": null
      }
    ]
  }
}
```


Response Fields:

Field	Description
orderId	Unique identifier of the order
orderNumber	Human-readable order number
buyer	Buyer account info (accountId, userName, email, firstName, lastName)
seller	Shop info (shopId, shopName, shopLogo, shopSlug)
productOrderStatus	PENDING_PAYMENT, PENDING_SHIPMENT, SHIPPED, DELIVERED, COMPLETED, CANCELLED, REFUNDED
deliveryStatus	PENDING, SHIPPED, IN_TRANSIT, DELIVERED, CONFIRMED, NOT_APPLICABLE
productOrderSource	DIRECT_PURCHASE, CART_PURCHASE, DIGITAL_PURCHASE, INSTALLMENT, GROUP_PURCHASE
items	Array of order items — see item fields below
items[].productType	`PHYSICAL` or `DIGITAL` — frontend uses this to show tracking UI vs download UI

Field	Description
items[].fileIds	List of file UUIDs for the item — populated only when `productType` is `DIGITAL`, `null` for physical. Use these with endpoint 14/15 to download files
subtotal	Sum of all item totals before shipping and tax
shippingFee	Shipping cost
tax	Tax amount
totalAmount	Final amount (subtotal + shipping + tax)
platformFee	Platform commission
sellerAmount	Amount seller receives after platform fee
currency	Currency code (TZS)
paymentMethod	Payment method used
amountPaid	Amount already paid
amountRemaining	Remaining balance (installment orders)
deliveryAddress	Shipping address
trackingNumber	Shipping tracking number (null until shipped)
carrier	Shipping carrier (null until shipped)
isDeliveryConfirmed	Whether buyer confirmed delivery
deliveryConfirmedAt	Timestamp of delivery confirmation (null if not confirmed)
orderedAt	Order creation timestamp
shippedAt	Shipping timestamp (null until shipped)
deliveredAt	Delivery timestamp (null until delivered)
cancelledAt	Cancellation timestamp (null if not cancelled)
cancellationReason	Reason for cancellation (null if not cancelled)
timeline	Ordered list of status steps — see Timeline Fields below
timeline[].status	Step identifier: `ORDER_PLACED`, `SHIPPED`, `DELIVERED`, `COMPLETED`, `FILES_AVAILABLE` (digital), `CANCELLED`, `DISPUTED`, `REFUNDED`
timeline[].label	Human-readable step label
timeline[].timestamp	When this step occurred (`null` if not yet reached)
timeline[].isCompleted	`true` if this step has been reached
timeline[].note	Optional context — shipping carrier + tracking on SHIPPED, cancellation reason on CANCELLED, "Confirmed by buyer" or "Auto-confirmed" on COMPLETED, `null` otherwise

Error Responses:

- `400 Bad Request`: Access denied — user is not buyer or seller of this order
- `401 Unauthorized`: Authentication required
- `404 Not Found`: Order not found

---

2. Get Order by Order Number

Purpose: Retrieve order details using the human-readable order number.

Endpoint: `GET` `{base}/number/{orderNumber}`

Access Level: □□ Protected (Buyer or Seller only)

Authentication: Bearer Token

Request Headers:

Header	Type	Required	Description
Authorization	string	Yes	Bearer token for authentication

Path Parameters:

Parameter	Type	Required	Description
orderNumber	string	Yes	Human-readable order number (e.g. ORD-2025-12345)

Response: Same structure as Get Order by ID.

Error Responses:

- `400 Bad Request`: Access denied — user is not buyer or seller of this order
- `401 Unauthorized`: Authentication required
- `404 Not Found`: Order not found

---

3. Get My Orders

Purpose: Retrieve all orders for the authenticated customer.

Endpoint: `GET` `{base}/my-orders`

Access Level: □□ Protected (Customer only)

Authentication: Bearer Token

Request Headers:

Header	Type	Required	Description
Authorization	string	Yes	Bearer token for authentication

Response JSON Sample:

```
{
  "success": true,
  "message": "Orders retrieved successfully",
  "data": [ ...array of order objects (same structure as endpoint 1)... ]
}
```

Error Responses:

- `401 Unauthorized`: Authentication required
- `404 Not Found`: User account not found

---

4. Get My Orders by Status

Purpose: Retrieve customer orders filtered by order status.

Endpoint: `GET` `{base}/my-orders/status/{status}`

Access Level: □□ Protected (Customer only)

Authentication: Bearer Token

Request Headers:

Header	Type	Required	Description
Authorization	string	Yes	Bearer token for authentication

Path Parameters:

Parameter	Type	Required	Description

status	enum	Yes	PENDING_PAYMENT, PENDING_SHIPMENT, SHIPPED, DELIVERED, COMPLETED, CANCELLED, REFUNDED

Response: Array of order objects (same structure as endpoint 1).

Error Responses:

- `400 Bad Request`: Invalid status value
- `401 Unauthorized`: Authentication required
- `404 Not Found`: User account not found

---

5. Get My Orders (Paginated)

Purpose: Retrieve customer orders with pagination.

Endpoint: `GET` `{base}/my-orders/paged`

Access Level: □□ Protected (Customer only)

Authentication: Bearer Token

Request Headers:

Header	Type	Required	Description
Authorization	string	Yes	Bearer token for authentication

Query Parameters:

Parameter	Type	Required	Default	Description
page	integer	No	1	Page number (1-based)
size	integer	No	10	Number of items per page

Response JSON Sample:

```
{
  "success": true,
  "message": "Orders retrieved successfully",
  "data": {
```

```
    "orders": [ "...order objects (same structure as endpoint 1)..." ],
    "currentPage": 1,
    "pageSize": 10,
    "totalElements": 25,
    "totalPages": 3,
    "hasNext": true,
    "hasPrevious": false,
    "isFirst": true,
    "isLast": false
  }
}
```

Error Responses:

- `401 Unauthorized`: Authentication required
- `404 Not Found`: User account not found

---

6. Get My Orders by Status (Paginated)

Purpose: Retrieve customer orders filtered by status with pagination.

Endpoint: `GET` `{base}/my-orders/status/{status}/paged`

Access Level: □□ Protected (Customer only)

Authentication: Bearer Token

Request Headers:

Header	Type	Required	Description
Authorization	string	Yes	Bearer token for authentication

Path Parameters:

Parameter	Type	Required	Description
status	enum	Yes	PENDING_PAYMENT, PENDING_SHIPMENT, SHIPPED, DELIVERED, COMPLETED, CANCELLED, REFUNDED

Query Parameters:

Parameter	Type	Required	Default	Description
page	integer	No	1	Page number (1-based)
size	integer	No	10	Number of items per page

Response: Same paginated structure as endpoint 5.

Error Responses:

- `400 Bad Request`: Invalid status value
- `401 Unauthorized`: Authentication required
- `404 Not Found`: User account not found

---

7. Get Shop Orders

Purpose: Retrieve all orders for a specific shop.

Endpoint: `GET` `{base}/shop/{shopId}/orders`

Access Level: □□ Protected (Shop Owner only)

Authentication: Bearer Token

Request Headers:

Header	Type	Required	Description
Authorization	string	Yes	Bearer token for authentication

Path Parameters:

Parameter	Type	Required	Description
shopId	UUID	Yes	Unique identifier of the shop

Response: Array of order objects (same structure as endpoint 1).

Error Responses:

- `400 Bad Request`: User is not the owner of this shop
- `401 Unauthorized`: Authentication required
- `404 Not Found`: Shop not found

---

8. Get Shop Orders by Status

Purpose: Retrieve shop orders filtered by order status.

Endpoint: `GET` `{base}/shop/{shopId}/orders/status/{status}`

Access Level: □□ Protected (Shop Owner only)

Authentication: Bearer Token

Request Headers:

Header	Type	Required	Description
Authorization	string	Yes	Bearer token for authentication

Path Parameters:

Parameter	Type	Required	Description
shopId	UUID	Yes	Unique identifier of the shop
status	enum	Yes	PENDING_PAYMENT, PENDING_SHIPMENT, SHIPPED, DELIVERED, COMPLETED, CANCELLED, REFUNDED

Response: Array of order objects (same structure as endpoint 1).

Error Responses:

- `400 Bad Request`: Invalid status value or user is not shop owner
- `401 Unauthorized`: Authentication required
- `404 Not Found`: Shop not found

---

9. Get Shop Orders (Paginated)

Purpose: Retrieve shop orders with pagination.

Endpoint: `GET` `{base}/shop/{shopId}/orders/paged`

Access Level: □□ Protected (Shop Owner only)

Authentication: Bearer Token

Request Headers:

Header	Type	Required	Description
Authorization	string	Yes	Bearer token for authentication

Path Parameters:

Parameter	Type	Required	Description
shopId	UUID	Yes	Unique identifier of the shop

Query Parameters:

Parameter	Type	Required	Default	Description
page	integer	No	1	Page number (1-based)
size	integer	No	10	Number of items per page

Response: Same paginated structure as endpoint 5.

Error Responses:

- `400 Bad Request`: User is not the owner of this shop
- `401 Unauthorized`: Authentication required
- `404 Not Found`: Shop not found

---

10. Get Shop Orders by Status (Paginated)

Purpose: Retrieve shop orders filtered by status with pagination.

Endpoint: `GET` `{base}/shop/{shopId}/orders/status/{status}/paged`

Access Level: □□ Protected (Shop Owner only)

Authentication: Bearer Token

Request Headers:

Header	Type	Required	Description
Authorization	string	Yes	Bearer token for authentication

Path Parameters:

Parameter	Type	Required	Description
shopId	UUID	Yes	Unique identifier of the shop
status	enum	Yes	PENDING_PAYMENT, PENDING_SHIPMENT, SHIPPED, DELIVERED, COMPLETED, CANCELLED, REFUNDED

Query Parameters:

Parameter	Type	Required	Default	Description
page	integer	No	1	Page number (1-based)
size	integer	No	10	Number of items per page

Response: Same paginated structure as endpoint 5.

Error Responses:

- `400 Bad Request`: Invalid status value or user is not shop owner
- `401 Unauthorized`: Authentication required
- `404 Not Found`: Shop not found

---

11. Mark Order as Shipped

Purpose: Seller marks an order as shipped. Generates a delivery confirmation code and sends it to the buyer.

Endpoint: `POST` `{base}/{orderId}/ship`

Access Level: □□ Protected (Shop Owner/Seller only)

Authentication: Bearer Token

Request Headers:

Header	Type	Required	Description
Authorization	string	Yes	Bearer token for authentication

Path Parameters:

Parameter	Type	Required	Description
orderId	UUID	Yes	Unique identifier of the order

Response JSON Sample:

```
{
  "success": true,
  "message": "Order marked as shipped",
  "data": {
    "orderId": "550e8400-e29b-41d4-a716-446655440000",
    "orderNumber": "ORD-2025-12345",
    "shippedAt": "2025-10-25T10:30:45",
    "message": "Order marked as shipped. Confirmation code sent to customer.",
    "confirmationCodeSent": true,
    "codeExpiresAt": "2025-11-24T10:30:45",
    "maxVerificationAttempts": 5
  }
}
```

Response Fields:

Field	Description
orderId	UUID of the shipped order
orderNumber	Human-readable order number
shippedAt	Timestamp when order was marked as shipped
message	Confirmation message
confirmationCodeSent	Whether confirmation code was sent to customer
codeExpiresAt	When the confirmation code expires (30 days from generation)
maxVerificationAttempts	Maximum number of code verification attempts allowed

Error Responses:

- `400 Bad Request`: Order is a digital order (does not require shipping), order status is not PENDING_SHIPMENT, or user is not the seller
- `401 Unauthorized`: Authentication required
- `404 Not Found`: Order not found

---

12. Confirm Delivery

Purpose: Customer confirms order delivery using the 6-digit confirmation code. Releases escrow to seller.

Endpoint: POST `{base}/{orderId}/confirm-delivery`

Access Level: Protected (Buyer/Customer only)

Authentication: Bearer Token

Request Headers:

Header	Type	Required	Description
Authorization	string	Yes	Bearer token for authentication
User-Agent	string	No	Device info for verification tracking
X-Forwarded-For	string	No	Client IP address (if behind proxy)
X-Real-IP	string	No	Real client IP address

Path Parameters:

Parameter	Type	Required	Description
orderId	UUID	Yes	Unique identifier of the order

Request JSON Sample:

```
{
  "confirmationCode": "123456"
}
```

Request Body Parameters:

Parameter	Type	Required	Description	Validation
confirmationCode	string	Yes	6-digit delivery confirmation code	Exactly 6 digits (0-9)

Response JSON Sample:

```
{
  "orderId": "550e8400-e29b-41d4-a716-446655440000",
  "orderNumber": "ORD-2025-12345",
  "deliveredAt": "2025-10-25T10:30:45",
  "confirmedAt": "2025-10-25T10:30:45",
  "escrowReleased": true,
  "sellerAmount": 166250.00,
  "currency": "TZS",
  "message": "Delivery confirmed successfully. Order completed!"
}
```


Note: This response is returned directly without the standard success envelope.

Response Fields:

Field	Description
orderId	UUID of the confirmed order
orderNumber	Human-readable order number
deliveredAt	Timestamp when order was marked as delivered
confirmedAt	Timestamp when delivery was confirmed
escrowReleased	Whether escrow funds were released to seller
sellerAmount	Amount released to seller after platform fee
currency	Currency code
message	Confirmation message

Error Responses:

- `400 Bad Request`: Order is a digital order (completed automatically, no confirmation needed), invalid confirmation code, order not SHIPPED, user is not buyer, max attempts exceeded, code expired, or escrow already released
- `401 Unauthorized`: Authentication required
- `404 Not Found`: Order not found or no active confirmation code
- `422 Unprocessable Entity`: Confirmation code format invalid

---

13. Regenerate Confirmation Code

Purpose: Customer requests a new delivery confirmation code if the previous one was lost or expired.

Endpoint: POST `{base}/{orderId}/regenerate-code`

Access Level: □□ Protected (Buyer/Customer only)

Authentication: Bearer Token

Request Headers:

Header	Type	Required	Description
Authorization	string	Yes	Bearer token for authentication

Path Parameters:

Parameter	Type	Required	Description
orderId	UUID	Yes	Unique identifier of the order

Response JSON Sample:

```
{
  "success": true,
  "message": "Confirmation code regenerated successfully",
  "data": {
    "orderId": "550e8400-e29b-41d4-a716-446655440000",
    "orderNumber": "ORD-2025-12345",
    "codeSent": true,
    "destination": "email",
    "codeExpiresAt": "2025-11-24T10:30:45",
    "maxAttempts": 5,
    "message": "New confirmation code sent to your email"
  }
}
```

Response Fields:

Field	Description
orderId	UUID of the order
orderNumber	Human-readable order number
codeSent	Whether new code was successfully sent
destination	Where the code was sent (`email`)
codeExpiresAt	When the new code expires (30 days from generation)
maxAttempts	Maximum number of verification attempts allowed

Field	Description
message	Confirmation message

Error Responses:

- `400 Bad Request`: Order is a digital order (does not use delivery confirmation codes), order status is not SHIPPED, user is not the buyer, or delivery already confirmed
- `401 Unauthorized`: Authentication required
- `404 Not Found`: Order not found

---

14. Get Digital Download URL

Purpose: Generates a presigned download URL for a digital file linked to an order. The URL expires in 5 minutes.

Endpoint: `GET` `{base}/{orderId}/downloads/{fileId}`

Access Level: □□ Protected (Buyer only)

Authentication: Bearer Token

Request Headers:

Header	Type	Required	Description
Authorization	string	Yes	Bearer token for authentication

Path Parameters:

Parameter	Type	Required	Description
orderId	UUID	Yes	Unique identifier of the order
fileId	UUID	Yes	Unique identifier of the digital file

Response JSON Sample:

```
{
  "success": true,
  "message": "Download URL generated — link expires in 5 minutes",
  "data": {
    "fileId": "abc12345-def6-7890-ghij-klmnopqrstuv",
```

```
    "fileName": "course-material.pdf",
    "downloadUrl": "https://storage.example.com/files/...",
    "expiresAt": "2025-10-25T10:35:45",
    "downloadsRemaining": 3,
    "downloadCount": 2
  }
}
```


Response Fields:

Field	Description
fileId	Unique identifier of the digital file
fileName	Name of the file
downloadUrl	Presigned URL for downloading the file (expires in 5 minutes)
expiresAt	Timestamp when the download URL expires
downloadsRemaining	Number of downloads remaining for this buyer
downloadCount	Number of times this file has been downloaded

Error Responses:

- `401 Unauthorized`: Authentication required
- `404 Not Found`: Order or file not found
- `422 Unprocessable Entity`: Download limit exceeded or access not permitted

---

15. List Order Downloads

Purpose: Returns all digital files the buyer has access to for a given order, including `fileId` needed to generate download URLs. Call this first before endpoint 14.

Endpoint: `GET` `{base}/{orderId}/downloads`

Access Level: Protected (Buyer only)

Authentication: Bearer Token

Request Headers:

Header	Type	Required	Description
Authorization	string	Yes	Bearer token for authentication

Path Parameters:

Parameter	Type	Required	Description
orderId	UUID	Yes	Unique identifier of the order

Response JSON Sample:


```
{
  "success": true,
  "message": "2 file(s) available for download",
  "data": [
    {
      "fileId": "f1a2b3c4-def5-6789-ghij-klmnopqrstuv",
      "fileName": "spring-boot-course.zip",
      "contentType": "application/zip",
      "fileSize": 524288000,
      "downloadCount": 1,
      "downloadsRemaining": 4,
      "accessExpiresAt": "2026-06-18T10:00:00",
      "canDownload": true
    },
    {
      "fileId": "a9b8c7d6-e5f4-3210-hijk-lmnopqrstuvw",
      "fileName": "bonus-resources.pdf",
      "contentType": "application/pdf",
      "fileSize": 2048000,
      "downloadCount": 0,
      "downloadsRemaining": 4,
      "accessExpiresAt": "2026-06-18T10:00:00",
      "canDownload": true
    }
  ]
}
```


Response Fields:

Field	Description
fileId	Use this in endpoint 14 to get the actual download URL
fileName	Display name of the file

Field	Description
contentType	MIME type of the file
fileSize	File size in bytes
downloadCount	How many times this buyer has downloaded this file
downloadsRemaining	Downloads left before cap is hit (null = unlimited)
accessExpiresAt	When this buyer's access to this file expires
canDownload	false if access is revoked, expired, or download cap reached

Error Responses:

- `401 Unauthorized`: Authentication required
- `404 Not Found`: Order not found or does not belong to this buyer
- `422 Unprocessable Entity`: Order has no digital files

Order Creation — How Orders Are Generated

Orders are never created manually. They are generated automatically when a checkout session moves to `PAYMENT_COMPLETED`. The system reads the session, applies grouping rules, and creates one or more orders depending on the cart contents and purchase type.

Order Grouping Rules

The grouping key is **(shop + product type)**. Two items end up in the same order only if they share both the same shop and the same product type.

Scenario	Result
Same shop, same type (both PHYSICAL)	1 order
Same shop, same type (both DIGITAL)	1 order
Same shop, different types (PHYSICAL + DIGITAL)	2 separate orders
Different shops, same type	1 order per shop
Different shops, different types	1 order per shop per type

Why split by type? Digital orders complete immediately (no shipping, escrow released on creation). Physical orders wait for seller shipment then buyer confirmation. Mixing them in one

order would make status tracking and escrow management impossible.

Scenario 1 — Direct Purchase (Buy Now)

Buyer clicks Buy Now on a single product. Always produces exactly one order.

Physical product:

```
Buyer → Buy Now → Payment
  → 1 order created (source: DIRECT_PURCHASE)
  → status: PENDING_SHIPMENT
  → escrow held until buyer confirms delivery
  → seller ships → buyer confirms with 6-digit code → escrow released → COMPLETED
```

Digital product:

```
Buyer → Buy Now → Payment
  → 1 order created (source: DIGITAL_PURCHASE)
  → status: COMPLETED immediately
  → escrow released immediately
  → DigitalDownloadAccess records created for all active files
  → buyer can download right away
```

Scenario 2 — Cart Purchase

Buyer checks out a cart with multiple items. The system groups by (shop, product type) and creates one order per group.

Example cart:

Item	Shop	Type
Wireless Headphones	TechStore	PHYSICAL
Spring Boot Course (PDF)	TechStore	DIGITAL
Running Shoes	SportShop	PHYSICAL

Result: 3 orders created

```
Order #1 → TechStore | PHYSICAL
  source: CART_PURCHASE
```

```
  status: PENDING_SHIPMENT
  shipping: split proportionally if multi-shop

Order #2 → TechStore | DIGITAL
  source: DIGITAL_PURCHASE
  status: COMPLETED immediately
  shipping: TZS 0
  → DigitalDownloadAccess created for Spring Boot Course files
  → buyer can download immediately

Order #3 → SportShop | PHYSICAL
  source: CART_PURCHASE
  status: PENDING_SHIPMENT
  shipping: split proportionally
```

Shipping split rule: If the cart has items from multiple shops, the total shipping cost is divided equally across the number of distinct shops. Each physical order gets its share.

Scenario 3 — Installment Purchase (IMMEDIATE fulfillment)

Buyer pays in installments but gets the product after the first payment.

Physical product:

```
First payment → order created (source: INSTALLMENT)
  → status: PENDING_SHIPMENT
  → seller ships after first payment
  → buyer confirms delivery → escrow released proportionally as payments come in

Remaining payments → collected without creating new orders
```

Digital product:

```
First payment → order created (source: INSTALLMENT → detected as DIGITAL_PURCHASE)
  → status: COMPLETED immediately
  → DigitalDownloadAccess created
  → buyer can download after first payment
```

```
Remaining payments → collected, no new order needed
```

Scenario 4 — Installment Purchase (AFTER_PAYMENT fulfillment)

Buyer pays all installments first, gets the product only after full payment.

Physical product:

```
First payment → no order created yet, agreement tracked only
...
Final payment → order created (source: INSTALLMENT)
  → status: PENDING_SHIPMENT
  → seller ships → buyer confirms → COMPLETED
```

Digital product:

```
First payment → no order created yet
...
Final payment → order created (source: INSTALLMENT → detected as DIGITAL_PURCHASE)
  → status: COMPLETED immediately
  → DigitalDownloadAccess created
  → buyer can download only after all installments are paid
```

Scenario 5 — Group Purchase

Multiple buyers join a group for a discounted price. When the group reaches its participant goal, an order is created for every participant simultaneously.

Physical product:

```
Group goal reached →
  For each participant:
    → 1 order created (source: GROUP_PURCHASE)
    → status: PENDING_SHIPMENT
    → seller ships to each buyer individually
    → each buyer confirms delivery independently
```

Digital product:

```
Group goal reached →
  For each participant:
    → 1 order created (source: GROUP_PURCHASE → detected as DIGITAL_PURCHASE)
    → status: COMPLETED immediately
    → DigitalDownloadAccess created per participant
    → all buyers can download simultaneously
```

Group metadata stored on each order: `groupInstanceId`, `groupPrice`, `regularPrice`, `savings`.

Digital Download Flow (after any purchase)

Once an order with source `DIGITAL_PURCHASE` is created, the fulfillment service creates a `DigitalDownloadAccess` record per file per buyer. These records enforce:

Rule	Configured by
Access expiry	`product.downloadExpiryDays` (default: 365 days)
Max downloads	`product.maxDownloadsPerBuyer` (null = unlimited)
Per-download URL TTL	5 minutes (hardcoded)

Frontend download flow:

Step 1 — List available files for an order:

```
GET api/v1/e-commerce/orders/{orderId}/downloads

Response:
[
  {
    "fileId": "f1a2b3c4-...",
    "fileName": "spring-boot-course.zip",
    "contentType": "application/zip",
    "fileSize": 524288000,
    "downloadCount": 0,
    "downloadsRemaining": 5,
    "accessExpiresAt": "2026-06-18T10:00:00",
    "canDownload": true
  }
]
```

Step 2 — Get a short-lived download link per file:

```
GET api/v1/e-commerce/orders/{orderId}/downloads/{fileId}

Response:
{
  "fileId": "f1a2b3c4-...",
  "fileName": "spring-boot-course.zip",
  "downloadUrl": "https://storage.../...?X-Amz-Expires=300&...",
  "expiresAt": "2026-05-19T11:05:00",
  "downloadsRemaining": 4,
  "downloadCount": 1
}
```

Step 3 — Buyer hits `downloadUrl` directly. The URL points to MinIO and expires in 5 minutes. Each call to Step 2 increments `downloadCount`.

Order Status Reference

Status	Applies to	Meaning
`PENDING_SHIPMENT`	Physical	Order paid, waiting for seller to ship
`SHIPPED`	Physical	Seller marked as shipped, waiting for buyer confirmation
`COMPLETED`	Both	Physical: buyer confirmed delivery. Digital: set immediately on creation
`CANCELLED`	Both	Order cancelled
`REFUNDED`	Both	Payment refunded

Delivery Status	Applies to	Meaning
`PENDING`	Physical	Not yet shipped
`IN_TRANSIT`	Physical	Seller marked as shipped
`CONFIRMED`	Physical	Buyer confirmed receipt
`NOT_APPLICABLE`	Digital	No physical delivery involved

Timeline Reference

The `timeline` field is embedded in every order detail response. It is a sequential list of steps representing the order's lifecycle. Steps not yet reached have `timestamp: null` and `isCompleted: false` — the frontend renders these as pending/greyed-out.

Physical order steps (DIRECT_PURCHASE, CART_PURCHASE, INSTALLMENT, GROUP_PURCHASE):

```
ORDER_PLACED → SHIPPED → DELIVERED → COMPLETED
```

Digital order steps (DIGITAL_PURCHASE):

```
ORDER_PLACED → FILES_AVAILABLE → COMPLETED
```

Terminal branches (replace remaining steps when reached):

```
CANCELLED  — appears after ORDER_PLACED if cancelled before shipping
DISPUTED   — appears after SHIPPED/FILES_AVAILABLE if buyer raises a dispute
REFUNDED   — appears after DISPUTED if resolved in buyer's favour
```

Step notes:

Step	Note value
`SHIPPED`	`"<Carrier> · <TrackingNumber>"` if tracking info is set, otherwise `null`
`CANCELLED`	Cancellation reason if provided, otherwise `null`
`COMPLETED`	`"Confirmed by buyer"` or `"Auto-confirmed"` depending on how it was confirmed
All others	`null`

Revision #11
Created 18 October 2025 20:34:25 by Admin Qbit
Updated 10 June 2026 12:41:48 by Admin Qbit