Event Booking Orders API

Author: Josh, Lead Backend Team
Last Updated: 2025-12-11
Version: v1.0

Base URL: `https://api.nexgate.com/api/v1`

Short Description: The Event Booking Orders API provides comprehensive booking management functionality for confirmed event ticket purchases in the Nexgate platform. This API enables users to view their booking details including JWT-signed QR codes for secure event entry, track multi-day event check-ins, retrieve complete booking history, and access event snapshots at time of booking. The system supports both single-day and multi-day events with per-day check-in tracking, automated ticket series generation, and comprehensive booking references.

Hints:

- **Auto-Creation**: Bookings created automatically after successful checkout payment
- **JWT Security**: QR codes are RSA-signed JWTs containing ticket and event data
- **Multi-Day Support**: Full support for multi-day events with per-day check-in tracking
- **Ticket Series**: Auto-generated unique series (e.g., "VIP-0001", "GENER-0042")
- **Event Snapshots**: Event details captured at booking time (immutable)
- **Booking Reference**: Short readable codes (e.g., "EVT-2025-A3F4")
- **Check-In Tracking**: Complete history with location, staff, and day information
- **Access Control**: Customers see own bookings, organizers see event bookings, admins see all
- **Notification Integration**: Auto-emails tickets to buyer and optionally to attendees
- **Virtual Events**: Includes virtual meeting links for online/hybrid events
- **Attendee Management**: Track tickets for buyer and other attendees separately

---

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": "2025-12-11T10:30:45",
  "data": {
    // Actual response data goes here
  }
}
```

Error Response Structure

```
{
  "success": false,
  "httpStatus": "BAD_REQUEST",
  "message": "Error description",
  "action_time": "2025-12-11T10:30:45",
  "data": "Error description"
}
```

HTTP Method Badge Standards

- **GET** - `GET` - Green (Read operations)

BookingOrderResponse Structure

This is the comprehensive response structure returned by booking endpoints:

```
{
  "bookingId": "550e8400-e29b-41d4-a716-446655440000",
  "bookingReference": "EVT-A3F4B21C",
  "status": "CONFIRMED",
  "event": {
    "eventId": "770e8400-e29b-41d4-a716-446655440002",
    "title": "East African Tech Summit 2025",
    "startDateTime": "2025-12-15T09:00:00",
    "endDateTime": "2025-12-17T18:00:00",
    "timezone": "Africa/Nairobi",
```

```
  "location": "KICC Nairobi, Harambee Avenue, Nairobi",
  "format": "HYBRID",
  "virtualDetails": {
    "platform": "ZOOM",
    "meetingUrl": "https://zoom.us/j/123456789",
    "meetingId": "123 456 789",
    "passcode": "summit2025",
    "additionalInstructions": "Join 5 minutes early for networking"
  }
},
"organizer": {
  "name": "TechEvents Kenya",
  "email": "organizer@techevents.ke",
  "phone": "+254712345678"
},
"customer": {
  "customerId": "660e8400-e29b-41d4-a716-446655440001",
  "name": "johndoe",
  "email": "john@example.com"
},
"tickets": [
  {
    "ticketInstanceId": "880e8400-e29b-41d4-a716-446655440010",
    "ticketTypeName": "VIP Pass",
    "ticketSeries": "VIP-0001",
    "price": 150.00,
    "qrCode": "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9...",
    "attendanceMode": "IN_PERSON",
    "attendee": {
      "name": "John Doe",
      "email": "john@example.com",
      "phone": "+255712345678"
    },
    "buyer": {
      "name": "John Doe",
      "email": "john@example.com"
    },
    "checkIns": [
      {
```

```
        "checkInTime": "2025-12-15T09:15:00+03:00",
        "checkInLocation": "Main Gate",
        "checkedInBy": "Scanner Operator 1",
        "dayName": "Day 1 - Opening Day",
        "scannerId": "SCANNER-001",
        "checkInMethod": "QR_SCAN"
      },
      {
        "checkInTime": "2025-12-16T08:45:00+03:00",
        "checkInLocation": "VIP Entrance",
        "checkedInBy": "Scanner Operator 2",
        "dayName": "Day 2 - Conference Day",
        "scannerId": "SCANNER-003",
        "checkInMethod": "QR_SCAN"
      }
    ],
    "hasBeenCheckedIn": true,
    "lastCheckedInAt": "2025-12-16T08:45:00+03:00",
    "lastCheckedInBy": "Scanner Operator 2",
    "lastCheckInLocation": "VIP Entrance",
    "lastCheckInDayName": "Day 2 - Conference Day",
    "status": "USED",
    "validFrom": "2025-12-15T09:00:00+03:00",
    "validUntil": "2025-12-17T18:00:00+03:00"
  },
  {
    "ticketInstanceId": "880e8400-e29b-41d4-a716-446655440011",
    "ticketTypeName": "General Admission",
    "ticketSeries": "GENER-0042",
    "price": 50.00,
    "qrCode": "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9...",
    "attendanceMode": "IN_PERSON",
    "attendee": {
      "name": "Jane Smith",
      "email": "jane@example.com",
      "phone": "+255723456789"
    },
    "buyer": {
      "name": "John Doe",
```

```
        "email": "john@example.com"
      },
      "checkIns": [],
      "hasBeenCheckedIn": false,
      "lastCheckedInAt": null,
      "lastCheckedInBy": null,
      "lastCheckInLocation": null,
      "lastCheckInDayName": null,
      "status": "ACTIVE",
      "validFrom": "2025-12-15T09:00:00+03:00",
      "validUntil": "2025-12-17T18:00:00+03:00"
    }
  ],
  "totalTickets": 2,
  "checkedInTicketsCount": 1,
  "subtotal": 200.00,
  "total": 200.00,
  "bookedAt": "2025-12-11T10:30:45",
  "cancelledAt": null
}
```


Response Field Descriptions

Root Level Fields

Field	Type	Description
bookingId	string (UUID)	Unique booking identifier
bookingReference	string	Short readable code (e.g., "EVT-A3F4B21C")
status	enum	Booking status: CONFIRMED, CANCELLED
event	object	Event snapshot at time of booking
organizer	object	Organizer details snapshot
customer	object	Customer (buyer) information
tickets	array	List of all booked tickets with check-in history
totalTickets	integer	Total number of tickets in booking
checkedInTicketsCount	integer	Count of tickets with at least one check-in

Field	Type	Description
subtotal	decimal	Subtotal amount
total	decimal	Total amount paid
bookedAt	string (LocalDateTime)	When booking was created
cancelledAt	string (LocalDateTime)	When booking was cancelled (null if active)

Event Snapshot Object

Field	Type	Description
eventId	string (UUID)	Event identifier
title	string	Event title at time of booking
startDateTime	string (LocalDateTime)	Event start date/time
endDateTime	string (LocalDateTime)	Event end date/time
timezone	string	IANA timezone (e.g., "Africa/Nairobi")
location	string	Full venue location (name + address)
format	string	Event format: IN_PERSON, ONLINE, HYBRID
virtualDetails	object	Virtual meeting details (for ONLINE/HYBRID)

Virtual Details Object

Field	Type	Description
platform	string	Platform: ZOOM, GOOGLE_MEET, MS_TEAMS, CUSTOM
meetingUrl	string	Full meeting URL
meetingId	string	Meeting ID (optional)
passcode	string	Meeting passcode (optional)
additionalInstructions	string	Extra instructions (optional)

Organizer Snapshot Object

Field	Type	Description
name	string	Organizer name at time of booking
email	string	Organizer email
phone	string	Organizer phone

Customer Info Object

Field	Type	Description
customerId	string (UUID)	Customer account ID
name	string	Customer username
email	string	Customer email

Booked Ticket Response Object

Field	Type	Description
ticketInstanceId	string (UUID)	Unique ticket instance ID
ticketTypeName	string	Ticket type name (e.g., "VIP Pass")
ticketSeries	string	Auto-generated series (e.g., "VIP-0001")
price	decimal	Ticket price
qrCode	string	JWT-signed QR code (very long string)
attendanceMode	string	IN_PERSON or ONLINE (for hybrid events)
attendee	object	Attendee information
buyer	object	Buyer information (who paid)
checkIns	array	Complete check-in history (all days)
hasBeenCheckedIn	boolean	True if checked in at least once
lastCheckedInAt	string (ZonedDateTime)	Most recent check-in time
lastCheckedInBy	string	Who performed last check-in
lastCheckInLocation	string	Location of last check-in
lastCheckInDayName	string	Day name of last check-in
status	enum	ACTIVE, USED, CANCELLED
validFrom	string (ZonedDateTime)	Ticket valid from
validUntil	string (ZonedDateTime)	Ticket valid until

Attendee/Buyer Info Object

Field	Type	Description
name	string	Person's name
email	string	Person's email
phone	string	Person's phone (attendee only)

Check-In Record Object

Field	Type	Description
checkInTime	string (ZonedDateTime)	When check-in occurred
checkInLocation	string	Where (e.g., "Main Gate", "VIP Entrance")
checkedInBy	string	Staff/scanner operator name
dayName	string	Event day (e.g., "Day 1", "Day 2 - Saturday")
scannerId	string	Scanner device ID
checkInMethod	string	Method: QR_SCAN (default), MANUAL, NFC

BookingOrderSummaryResponse Structure

Lightweight response for listing bookings:

```
{
  "bookingId": "550e8400-e29b-41d4-a716-446655440000",
  "bookingReference": "EVT-A3F4B21C",
  "status": "CONFIRMED",
  "eventTitle": "East African Tech Summit 2025",
  "eventStartDateTime": "2025-12-15T09:00:00",
  "eventLocation": "KICC Nairobi, Harambee Avenue, Nairobi",
  "totalTickets": 2,
  "checkedInTickets": 1,
  "total": 200.00,
  "bookedAt": "2025-12-11T10:30:45"
}
```

Endpoints

1. Get Booking by ID

Purpose: Retrieve complete booking details including all tickets and check-in history

Endpoint: `GET` `{base_url}/e-events/booking-orders/{bookingId}`

Access Level: □□ Protected (Booking Owner, Event Organizer, or Admin)

Authentication: Bearer Token

Request Headers:

Header	Type	Required	Description
Authorization	string	Yes	Bearer token (format: `Bearer <token>`)

Path Parameters:

Parameter	Type	Required	Description
bookingId	string (UUID)	Yes	Booking order ID

Success Response: Returns complete BookingOrderResponse structure

Success Response Message: "Booking retrieved successfully"

HTTP Status Code: 200 OK

Access Rules:

- **Customers**: Can view their own bookings
- **Organizers**: Can view bookings for their events
- **Admins** (SUPER_ADMIN, STAFF_ADMIN): Can view any booking

Behavior:

- Returns complete booking details with all tickets
- Includes event snapshot (as it was at booking time)
- Shows full check-in history for multi-day events
- Includes JWT-signed QR codes for all tickets
- Virtual details included for ONLINE/HYBRID events

Use Cases:

- Customer viewing booking confirmation
- Customer retrieving QR codes for event entry

- Organizer checking booking details
- Admin reviewing booking for support

Standard Error Types:

- `401 UNAUTHORIZED`: Authentication issues
- `403 FORBIDDEN`: User doesn't have permission to view this booking
- `404 NOT_FOUND`: Booking not found

Error Response Examples:

Booking Not Found (404):

```
{
  "success": false,
  "httpStatus": "NOT_FOUND",
  "message": "Booking not found: 550e8400-e29b-41d4-a716-446655440000",
  "action_time": "2025-12-11T10:30:45",
  "data": "Booking not found: 550e8400-e29b-41d4-a716-446655440000"
}
```

Access Denied (403):

```
{
  "success": false,
  "httpStatus": "FORBIDDEN",
  "message": "You don't have permission to view this booking",
  "action_time": "2025-12-11T10:30:45",
  "data": "You don't have permission to view this booking"
}
```

---

2. Get My Bookings

Purpose: Retrieve all bookings for the authenticated user

Endpoint: `GET` `{base_url}/e-events/booking-orders/my-bookings`

Access Level: □□ Protected (Authenticated Users)

Authentication: Bearer Token

Request Headers:

Header	Type	Required	Description
Authorization	string	Yes	Bearer token (format: `Bearer <token>`)

Success Response: Returns array of BookingOrderSummaryResponse

Success Response Message: "Bookings retrieved successfully"

Success Response JSON Sample:


```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Bookings retrieved successfully",
  "action_time": "2025-12-11T10:30:45",
  "data": [
    {
      "bookingId": "550e8400-e29b-41d4-a716-446655440000",
      "bookingReference": "EVT-A3F4B21C",
      "status": "CONFIRMED",
      "eventTitle": "East African Tech Summit 2025",
      "eventStartDateTime": "2025-12-15T09:00:00",
      "eventLocation": "KICC Nairobi, Harambee Avenue, Nairobi",
      "totalTickets": 2,
      "checkedInTickets": 1,
      "total": 200.00,
      "bookedAt": "2025-12-11T10:30:45"
    },
    {
      "bookingId": "550e8400-e29b-41d4-a716-446655440001",
      "bookingReference": "EVT-B5D2E12F",
      "status": "CONFIRMED",
      "eventTitle": "Dar es Salaam Food Festival",
      "eventStartDateTime": "2025-12-20T11:00:00",
      "eventLocation": "Mlimani City, Sam Nujoma Road, Dar es Salaam",
      "totalTickets": 4,
      "checkedInTickets": 0,
      "total": 150.00,
      "bookedAt": "2025-12-10T14:20:30"
    }
  ]
```

```
}
```

HTTP Status Code: 200 OK

Sorting: Bookings sorted by `bookedAt` descending (newest first)

Behavior:

- Returns lightweight summary for all user's bookings
- Sorted newest first
- Includes check-in progress (checkedInTickets / totalTickets)
- Shows upcoming and past events

Use Cases:

- User viewing booking history
- Dashboard showing all purchases
- Finding booking for upcoming event
- Checking past event attendance

Standard Error Types:

- `401 UNAUTHORIZED`: Authentication issues
- `404 NOT_FOUND`: User not found/authenticated

---

3. Download Ticket PDF

Purpose: Download the ticket PDF for a specific ticket belonging to the authenticated user

Endpoint: `GET` `{base_url}/e-events/booking-orders/tickets/{ticketId}/pdf`

Access Level: □□ Protected (Authenticated Users)

Authentication: Bearer Token

Request Headers:

Header	Type	Required	Description
Authorization	string	Yes	Bearer token (format: `Bearer <token>`)

Path Parameters:

Parameter	Type	Required	Description
ticketId	UUID	Yes	The unique ID of the ticket to download

Query Parameters:

Parameter	Type	Required	Default	Description
mode	string	No	`download`	Controls how the PDF is served. `download` forces a file save dialog. `inline` opens the PDF directly in the browser.

Mode Values:

Value	Content-Disposition	Behaviour
`download`	`attachment; filename="ticket-{series}.pdf"`	Browser prompts user to save the file
`inline`	`inline; filename="ticket-{series}.pdf"`	PDF opens directly in the browser tab

Success Response: Returns a binary PDF file

Success Response Headers:

Header	Value
Content-Type	`application/pdf`
Content-Disposition	`attachment; filename="ticket-{series}.pdf"` *(or `inline` depending on mode)*

HTTP Status Code: 200 OK

Behavior:

- Verifies the ticket belongs to the authenticated user before generating the PDF
- Generates the ticket PDF on-the-fly using the event's SVG template
- The PDF contains event details, attendee name, seat/series, QR code for check-in, and ticket number
- QR code encodes a signed JWT token used by gate staff to verify the ticket at entry
- File name in the `Content-Disposition` header uses the ticket's series (e.g. `ticket-VIP-0033.pdf`)

Use Cases:

- User downloading their ticket after booking
- Re-downloading a lost or deleted ticket
- Saving ticket to device for offline access at the event

Standard Error Types:

Code	Type	Description
401	`UNAUTHORIZED`	Missing or invalid Bearer token
403	`FORBIDDEN`	Ticket does not belong to the authenticated user
404	`NOT_FOUND`	Ticket with the given ID not found
500	`INTERNAL_SERVER_ERROR`	PDF generation failed

Booking Status Lifecycle

Status Descriptions

Status	Description	Can Cancel?	Tickets Valid?
CONFIRMED	Booking active and tickets valid	Future (not implemented)	Yes
CANCELLED	Booking cancelled (placeholder)	N/A	No

Current Implementation:

- All bookings created as CONFIRMED
- Cancellation not yet implemented (status exists for future)
- Tickets remain ACTIVE after booking until checked in

Ticket Instance Status

Status Lifecycle

Status	Description	Can Check In?
ACTIVE	Ticket ready for use	Yes

Status	Description	Can Check In?
USED	Ticket checked in	Yes (for multi-day)
CANCELLED	Ticket cancelled	No

Multi-Day Events:

- Ticket status remains ACTIVE even after first check-in
- Can check in multiple times (once per day)
- Status changes to USED after all expected check-ins
- Each check-in tracked separately in checkIns array

Ticket Series Generation

How Ticket Series Work

Format: `{TICKET_CODE}-{COUNTER}`

Examples:

- VIP Pass → `VIP-0001`, `VIP-0002`, `VIP-0003`
- General Admission → `GENER-0001`, `GENER-0002`
- Early Bird → `EARLY-0001`, `EARLY-0002`

Ticket Code Extraction:

1. Take first 5 characters of ticket type name
2. Remove spaces
3. Convert to uppercase
4. Fallback to "TICK" if name empty

Counter System:

- Separate counter per ticket type (stored in DB)
- Pessimistic locking prevents duplicates
- Counter never resets (unique forever)
- Increments with each ticket created

Example Flow:

```
Ticket Type: "VIP Pass"
Counter: 0
```

```
First Booking (3 tickets):
- VIP-0001 (counter now 1)
- VIP-0002 (counter now 2)
- VIP-0003 (counter now 3)

Second Booking (2 tickets):
- VIP-0004 (counter now 4)
- VIP-0005 (counter now 5)
```

Why Series Numbers?

- Easy reference for staff
- Quick ticket identification
- Helps track capacity sold
- Professional appearance on PDF tickets

JWT-Signed QR Codes

Security Architecture

What is the QR Code?

- Not a random string
- Full JWT (JSON Web Token)
- Signed with event's RSA private key
- Contains complete ticket and event data
- Cannot be forged or tampered with

JWT Structure:

```
Header.Payload.Signature

eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.
eyJ0aWNrZXRJbnN0YW5jZUlkIjoiODgw...
[RSA signature]
```

JWT Payload Contains:

- ticketInstanceId (UUID)

- ticketTypeId (UUID)
- ticketTypeName (string)
- ticketSeries (string)
- eventId (UUID)
- eventName (string)
- eventStartDateTime (timestamp)
- attendeeName (string)
- attendeeEmail (string)
- attendeePhone (string)
- attendanceMode (IN_PERSON/ONLINE)
- bookingReference (string)
- **eventSchedules** (array of days for multi-day events)
- validFrom (timestamp)
- validUntil (timestamp)

Multi-Day Event Schedules:

```
"eventSchedules": [
  {
    "dayName": "Day 1 - Opening Day",
    "startDateTime": "2025-12-15T09:00:00+03:00",
    "endDateTime": "2025-12-15T18:00:00+03:00",
    "description": "Conference Opening & Keynotes"
  },
  {
    "dayName": "Day 2 - Conference Day",
    "startDateTime": "2025-12-16T09:00:00+03:00",
    "endDateTime": "2025-12-16T18:00:00+03:00",
    "description": "Technical Sessions & Workshops"
  },
  {
    "dayName": "Day 3 - Closing Day",
    "startDateTime": "2025-12-17T09:00:00+03:00",
    "endDateTime": "2025-12-17T18:00:00+03:00",
    "description": "Finals & Networking"
  }
]
```

RSA Key Pair:

- Generated when event is published (2048-bit)
- Private key: Stored securely in database (for signing)
- Public key: Available to scanners (for verification)

- Cannot derive private key from public key

Verification Process (Scanner App):

1. Scanner reads QR code
2. Extracts JWT token
3. Fetches event's public key
4. Verifies JWT signature
5. Checks expiry and validity
6. Checks if already checked in for this day
7. Records check-in if valid

Why JWT over Simple Codes?

- **Cannot be forged**: Requires private key
- **Contains all data**: No database lookup needed
- **Offline capable**: Scanner can verify without internet
- **Tamper-proof**: Any modification breaks signature
- **Time-limited**: validFrom/validUntil built in
- **Day tracking**: Multi-day schedules embedded

Multi-Day Event Check-Ins

How Multi-Day Check-Ins Work

Single-Day Event:

- One check-in expected
- After check-in, status changes to USED
- Simple entry tracking

Multi-Day Event (e.g., 3-day festival):

- Multiple check-ins allowed (one per day)
- Each day tracked separately
- Ticket remains ACTIVE after first day
- Full history stored in `checkIns` array

Check-In Record Fields:

- **checkInTime**: When check-in happened
- **checkInLocation**: Where (e.g., "Main Gate", "VIP Entrance")
- **checkedInBy**: Staff/scanner operator name

- **dayName**: Which day (must match eventSchedules)
- **scannerId**: Scanner device ID
- **checkInMethod**: QR_SCAN, MANUAL, NFC (default: QR_SCAN)

Example: 3-Day Festival

Day 1 (Friday):

```
{
  "checkInTime": "2025-12-15T18:30:00+03:00",
  "checkInLocation": "Main Gate",
  "checkedInBy": "Staff Member 1",
  "dayName": "Day 1 - Friday Night",
  "scannerId": "SCANNER-001",
  "checkInMethod": "QR_SCAN"
}
```

Day 2 (Saturday):

```
{
  "checkInTime": "2025-12-16T14:15:00+03:00",
  "checkInLocation": "VIP Entrance",
  "checkedInBy": "Staff Member 2",
  "dayName": "Day 2 - Saturday",
  "scannerId": "SCANNER-003",
  "checkInMethod": "QR_SCAN"
}
```

Day 3 (Sunday):

```
{
  "checkInTime": "2025-12-17T12:00:00+03:00",
  "checkInLocation": "Main Gate",
  "checkedInBy": "Staff Member 1",
  "dayName": "Day 3 - Sunday",
  "scannerId": "SCANNER-001",
  "checkInMethod": "QR_SCAN"
}
```

Validation Rules:

- `dayName` must match one from JWT's eventSchedules

- Cannot check in twice on same day
- Can check in on different days
- Check-in time must be within day's start/end

Benefits:

- Accurate attendance tracking per day
- Prevents duplicate same-day entries
- Allows organizers to see daily attendance
- Supports flexible entry (don't need all days)

Event Snapshots

Why Snapshots?

Problem: Event details can change after booking

- Event renamed
- Time changed
- Venue moved
- Organizer updated contact

Solution: Capture event state at booking time

Snapshot Fields (Immutable after booking):

- eventTitle
- eventStartDateTime
- eventEndDateTime
- eventTimezone
- eventLocation (full venue details)
- eventFormat
- virtualDetails (for online/hybrid)
- organizerName
- organizerEmail
- organizerPhone

Benefits:

- **PDF Tickets**: Generate correct ticket with original details
- **Legal Record**: What customer actually purchased
- **Email Confirmations**: Show accurate booking details
- **Refund Logic**: Reference original event details

- **Audit Trail**: Track what changed vs. what was booked

Example Scenario:

1. User books ticket for "Tech Summit" at KICC
2. Snapshot created: "Tech Summit", "KICC Nairobi"
3. Organizer renames event to "African Tech Summit"
4. Organizer moves venue to "Safari Park Hotel"
5. User's booking still shows original: "Tech Summit at KICC"
6. This is what they paid for (legal protection)

Booking Creation Flow

Automatic Creation After Payment

Trigger: EventPaymentCompletedListener fires after successful payment

Steps:

1. **Fetch Entities**:
   - Get EventCheckoutSession
   - Get Event details
   - Get TicketType details
2. **Generate Booking Reference**:
   - Format: `EVT-{8-char-UUID}`
   - Example: `EVT-A3F4B21C`
   - Unique and readable
3. **Create Ticket Instances**:
   - For buyer's tickets (ticketsForMe count)
   - For each other attendee (their quantity)
   - Each ticket gets unique series number
   - Each ticket assigned to correct attendee
4. **Generate JWT QR Codes**:
   - Create JWT payload with ticket data
   - Include event schedules for multi-day
   - Sign with event's RSA private key
   - Set as ticket's qrCode field
5. **Snapshot Event Details**:
   - Capture current event title, times, location
   - Capture organizer details
   - Store in booking (immutable)
6. **Save Booking**:
   - Create EventBookingOrderEntity

- Save to database with all tickets
- Link to checkout session

7. **Update Checkout Session**:
   - Set createdBookingOrderId
   - Mark as COMPLETED
8. **Publish Event**:
   - Fire BookingCreatedEvent
   - Triggers notifications
9. **Send Notifications**:
   - Email buyer confirmation with tickets
   - Optionally email other attendees their tickets
   - Notify event organizer of new booking

Transaction Safety:

- Entire flow in single transaction
- If any step fails, everything rolls back
- No partial bookings created

Timing:

- Happens immediately after payment success
- Usually completes in < 2 seconds
- User gets instant confirmation

Notification System Integration

Who Gets Notified?

1. Buyer (ALWAYS):

- Receives booking confirmation
- Gets ALL tickets (if sendTicketsToAttendees = false)
- Gets only THEIR tickets (if sendTicketsToAttendees = true)
- Channels: EMAIL, SMS, PUSH, IN_APP
- Priority: HIGH

2. Other Attendees (CONDITIONAL):

- Only if sendTicketsToAttendees = true
- Each attendee gets ONLY their tickets
- Channels: EMAIL, SMS
- Priority: HIGH

- Grouped by email (all tickets for same person)

3. Event Organizer (ALWAYS):

- Notified of new booking
- Gets booking summary (not tickets)
- Channels: EMAIL, IN_APP
- Priority: NORMAL

Notification Content

Buyer Confirmation:

- Booking reference
- Event title and date
- Number of tickets
- QR codes attached
- Event location/virtual link
- Instructions for entry

Attendee Ticket Email:

- "You've received tickets from [Buyer Name]"
- Event details
- Their specific QR codes
- Entry instructions

Organizer Alert:

- New booking notification
- Buyer name and email
- Number of tickets sold
- Total amount
- Booking reference for lookup

QR Code Delivery

sendTicketsToAttendees = false:

```
Buyer receives:
- Ticket 1 QR (for self)
- Ticket 2 QR (for self)
- Ticket 3 QR (for Jane)
- Ticket 4 QR (for Jane)
```

```
- Ticket 5 QR (for Bob)

Buyer must forward to Jane and Bob manually
```

sendTicketsToAttendees = true:

```
Buyer receives:
- Ticket 1 QR (for self)
- Ticket 2 QR (for self)

Jane receives:
- Ticket 3 QR (for Jane)
- Ticket 4 QR (for Jane)

Bob receives:
- Ticket 5 QR (for Bob)

Everyone gets their own tickets directly
```

Access Control Rules

Who Can View Bookings?

1. Booking Owner (Customer):

- Can view all their own bookings
- GET /booking-orders/my-bookings (all)
- GET /booking-orders/{bookingId} (specific)

2. Event Organizer:

- Can view bookings for their events
- Useful for checking attendance
- Verifying purchases
- Customer support

3. Platform Admins:

- Roles: SUPER_ADMIN, STAFF_ADMIN
- Can view any booking

- For support and dispute resolution

4. Others:

- Cannot view bookings they don't own
- Cannot view other users' bookings
- 403 Forbidden error

Virtual Event Support

Virtual Details Structure

Included for:

- ONLINE events (only virtual)
- HYBRID events (both physical + virtual)

Fields:

- **platform**: ZOOM, GOOGLE_MEET, MS_TEAMS, CUSTOM
- **meetingUrl**: Full meeting URL (required)
- **meetingId**: Meeting ID if applicable
- **passcode**: Meeting passcode if required
- **additionalInstructions**: Extra guidance

Example (Zoom):

```
{
  "platform": "ZOOM",
  "meetingUrl": "https://zoom.us/j/123456789?pwd=abc123",
  "meetingId": "123 456 789",
  "passcode": "summit2025",
  "additionalInstructions": "Please join 5 minutes early for a smooth start"
}
```

Example (Custom):

```
{
  "platform": "CUSTOM",
  "meetingUrl": "https://customplatform.com/event/12345",
  "meetingId": null,
```

```
  "passcode": null,
  "additionalInstructions": "Use your booking email to log in"
}
```

Delivery:

- Included in booking response
- Sent in confirmation email
- Sent in ticket emails
- Available before event starts

Hybrid Events:

- Virtual details + physical venue
- Attendees choose format via ticket type
- IN_PERSON tickets: Physical entry
- ONLINE tickets: Virtual access
- Both in same booking possible

Date/Time Formats

DateTime Standards

LocalDateTime (No timezone):

- Format: `YYYY-MM-DDTHH:mm:ss`
- Example: `2025-12-15T09:00:00`
- Used for: bookedAt, cancelledAt, event snapshots

ZonedDateTime (With timezone):

- Format: `YYYY-MM-DDTHH:mm:ss±HH:mm`
- Example: `2025-12-15T09:00:00+03:00`
- Used for: validFrom, validUntil, checkInTime

Why Different Formats?

LocalDateTime:

- Server-local time
- Audit timestamps
- No conversion needed
- Simple comparison

ZonedDateTime:

- User-aware time
- Ticket validity
- Check-in times
- Handles timezone correctly

Example:

```
Event in Nairobi (UTC+3):
startDateTime: 2025-12-15T09:00:00+03:00

Ticket validFrom: 2025-12-15T09:00:00+03:00
User in London sees: 2025-12-15T06:00:00+00:00 ✓
User in New York sees: 2025-12-15T01:00:00-05:00 ✓
```

Validation Rules Summary

Booking Retrieval Validation

Access Validation:

- □ User must be authenticated
- □ Booking must exist
- □ User must be: owner OR organizer OR admin
- □ Other users cannot access

Data Validation:

- □ bookingId must be valid UUID
- □ Booking must not be deleted (soft delete support)

Quick Reference Guide

Common HTTP Status Codes

- `200 OK`: Successful request
- `401 Unauthorized`: Authentication required/failed

- `403 Forbidden`: User doesn't have permission
- `404 Not Found`: Booking not found

Booking Reference Format

- Pattern: `EVT-{8-char-UUID}`
- Examples: `EVT-A3F4B21C`, `EVT-F2D5E891`
- Unique and readable
- Easy for customer service

Ticket Series Format

- Pattern: `{CODE}-{COUNTER}`
- Examples: `VIP-0001`, `GENER-0042`, `EARLY-0123`
- Code: First 5 chars of ticket name
- Counter: Unique incrementing number

QR Code Format

- Type: RSA-signed JWT
- Length: ~1000-2000 characters
- Contains: Complete ticket + event data
- Cannot be: Forged, tampered, reused (tracked)

Data Format Standards

- **Dates**: LocalDateTime for timestamps, ZonedDateTime for validity
- **IDs**: UUID format
- **Money**: Decimal with 2 decimals (150.00)
- **Currency**: TZS (Tanzanian Shilling)
- **Timezone**: IANA format (Africa/Nairobi)

Booking Flow Checklist

1. □ User completes checkout payment
2. □ Payment listener triggers
3. □ System generates booking reference
4. □ System creates ticket instances
5. □ System generates JWT QR codes
6. □ System snapshots event details
7. □ System saves booking order
8. □ System publishes booking event

9. □ System sends email notifications
10. □ User receives confirmation + QR codes

Best Practices

For Users:

- Save QR codes to phone immediately
- Screenshot confirmation email
- Arrive early for check-in
- One QR per person at gate

For Developers:

- Always show booking reference prominently
- Display QR codes large enough to scan
- Cache booking details locally
- Handle offline QR display
- Show check-in history clearly
- Support PDF ticket generation

For Organizers:

- Provide clear entry instructions
- Test scanner apps before event
- Have backup manual verification
- Track attendance by day
- Monitor check-in progress live

Error Handling Tips

- **403 Forbidden**: Show "This booking belongs to another user"
- **404 Not Found**: Suggest "Check your email for correct reference"
- **Network Error**: Show cached booking if available
- **QR Load Failure**: Provide booking reference as backup

Common Mistakes to Avoid

□ Showing other users' bookings
□ Not displaying virtual details for ONLINE events
□ Forgetting to show multi-day check-in history
□ Not caching QR codes for offline use
□ Displaying mutable event data instead of snapshot
□ Not grouping attendees' tickets properly

- Ignoring timezone in validity checks

Additional Notes

Future Enhancements (Not Yet Implemented)

Booking Cancellation:

- Status: CANCELLED exists but not functional
- Future: Refund logic + ticket release
- Future: Cancellation deadline rules
- Future: Partial cancellations

Ticket Transfers:

- Currently: Tickets assigned at booking
- Future: Transfer ticket to another person
- Future: Resale marketplace
- Future: Name change requests

Check-In API:

- Currently: Manual check-ins not via API
- Future: Scanner app endpoints
- Future: QR verification endpoint
- Future: Manual check-in endpoint

Booking Modifications:

- Currently: Cannot modify after creation
- Future: Add more tickets
- Future: Change attendee names
- Future: Upgrade ticket types

Integration Points

Payment System:

- Booking created after payment success
- Linked via checkoutSessionId
- Payment status determines creation

Notification System:

- BookingCreatedEvent triggers emails
- Async processing via listeners
- Multiple notification channels

Escrow System:

- Payments held until event completion
- Booking tracks payment via checkout session
- Organizer paid after event

Scanner Apps:

- Will consume booking data
- Verify JWT QR codes
- Record check-ins
- Update ticket status

Conclusion

The Event Booking Orders API provides comprehensive booking management with:

- **Security**: JWT-signed QR codes prevent fraud
- **Flexibility**: Multi-day events with per-day tracking
- **Reliability**: Event snapshots preserve booking details
- **Scalability**: Unique ticket series with DB counters
- **User Experience**: Clear references and notifications
- **Multi-Attendee**: Support for group bookings
- **Virtual Events**: Full online/hybrid event support
- **Access Control**: Proper permissions for all parties

Revision #3
Created 11 December 2025 10:10:31 by Admin Qbit
Updated 26 February 2026 09:38:52 by Admin Qbit