Ticket Management API

Author: Josh S. Sakweli, Backend Lead Team **Last Updated**: 2026-02-20 **Version**: v1.2

Base URL: `https://your-api-domain.com/api/v1/e-events/tickets`

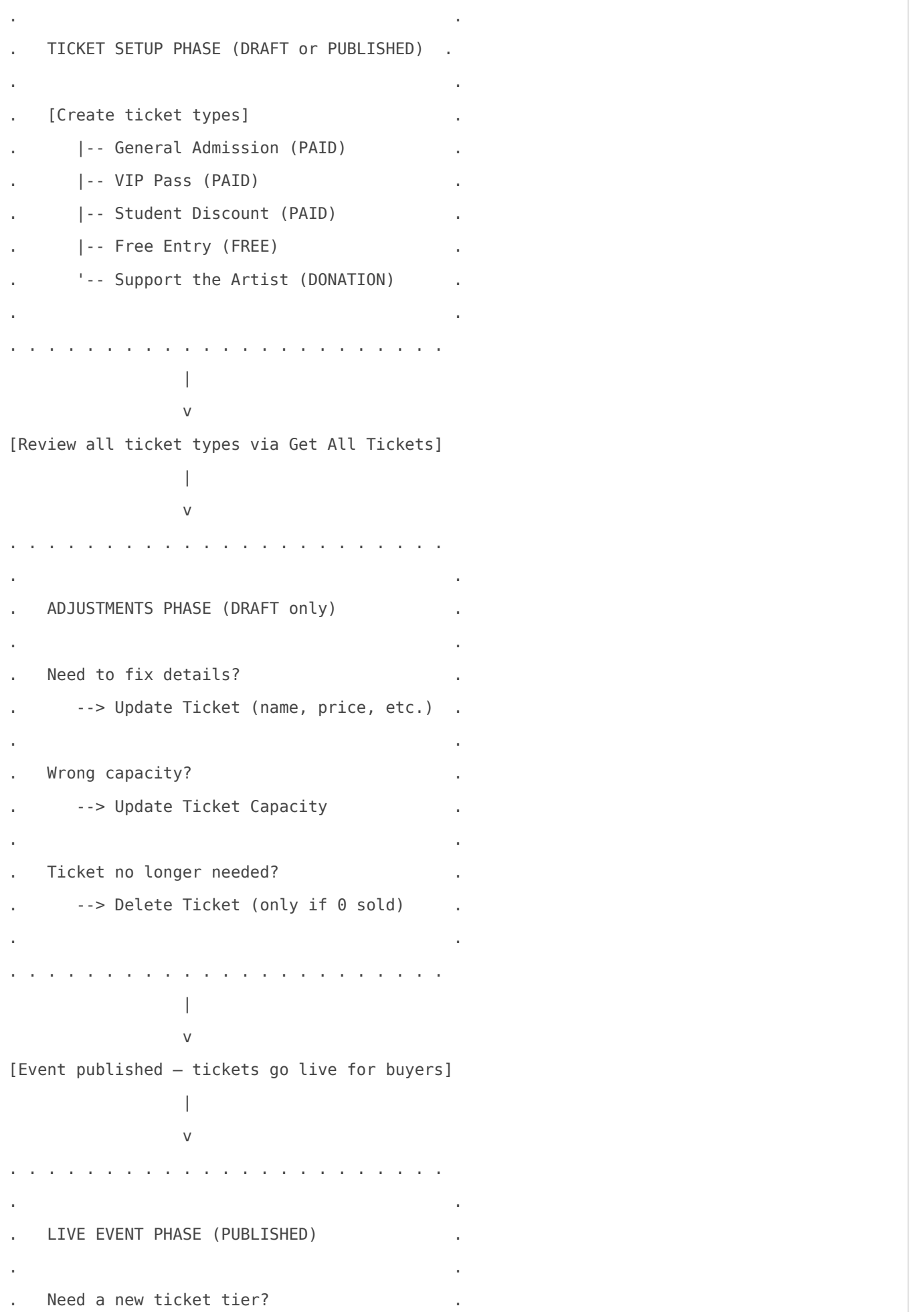
Short Description: The Ticket Management API allows event organizers to define and manage ticket types for their events on NextGate. Organizers can create multiple ticket tiers (e.g. VIP, Early Bird, General Admission), control pricing, capacity, sales periods, visibility, and sales channels. Tickets are scoped to a specific event and follow the event's lifecycle from DRAFT through to PUBLISHED.

Hints:

- Tickets can be created while the event is in **DRAFT or PUBLISHED** status. Full edits (name, price, type, etc.) are only allowed in **DRAFT**. After publishing, use the dedicated published-ticket endpoint for limited updates.
- For **HYBRID** events, you must create at least one `IN_PERSON` ticket and one `ONLINE` ticket before the event can be published.
- Ticket names must be unique per event per attendance mode — you cannot have two `IN_PERSON` tickets both named "VIP Pass" on the same event.
- **DONATION** tickets are restricted to `ONLINE_ONLY` sales channel and a maximum of 1 ticket per order and per user. They have no fixed price — the buyer freely chooses their donation amount at checkout.
- The ticket sales window must fall **within the event's registration window**. Sales cannot start before registration opens or end after registration closes.
- The minimum gap between `salesStartDateTime` and `salesEndDateTime` is **30 minutes**.
- Soft deletion is used — a ticket can only be deleted if zero tickets have been sold. Otherwise, close it using the status endpoint.
- All datetimes must be in **ISO 8601 / ZonedDateTime format** (e.g. `2025-08-10T09:00:00+03:00`).

User Journey

```
[Organizer creates event in DRAFT status]
                  |
                  | (Event must be DRAFT or PUBLISHED for ticket work)
                  v
. . . . . . . . . . . . . . . . . . . . . . . . .
```


TICKET SETUP PHASE (DRAFT or PUBLISHED)
[Create ticket types]
|-- General Admission (PAID)
|-- VIP Pass (PAID)
|-- Student Discount (PAID)
|-- Free Entry (FREE)
'-- Support the Artist (DONATION)
[Review all ticket types via Get All Tickets]
ADJUSTMENTS PHASE (DRAFT only)
Need to fix details?
--> Update Ticket (name, price, etc.)
Wrong capacity?
--> Update Ticket Capacity
Ticket no longer needed?
--> Delete Ticket (only if 0 sold)
[Event published — tickets go live for buyers]
LIVE EVENT PHASE (PUBLISHED)
Need a new ticket tier?

```
.      --> Create Ticket (allowed on PUBLISHED).
.                                              .
.   Need to adjust visibility/status?          .
.      --> Update Published Ticket             .
.                                              .
.   Need to shift the sales window?            .
.      --> Update Sales Window                 .
.                                              .
.   Ticket sells out?                          .
.      --> System auto-sets SOLD_OUT           .
.      --> Organizer can increase capacity     .
.          to reactivate it                    .
.                                              .
.   Want to pause sales temporarily?           .
.      --> Update Status to INACTIVE           .
.                                              .
.   Want to stop sales permanently?            .
.      --> Update Status to CLOSED             .
.                                              .
. . . . . . . . . . . . . . . . . . . . . . . .
```

Sales Window Rules

Ticket sales must respect three nested time windows:

```
Event:          [eventStartDateTime ———————————— eventEndDateTime]
Registration:         [registrationOpensAt ——— registrationClosesAt]
Ticket Sales:              [salesStartDateTime —— salesEndDateTime]
```

Rules enforced:

- `salesStartDateTime` cannot be in the past
- `salesEndDateTime` cannot be in the past
- `salesEndDateTime` must be after `salesStartDateTime`
- Minimum gap between sales start and end is **30 minutes**
- `salesStartDateTime` must be on or after `registrationOpensAt`
- `salesStartDateTime` cannot be after `registrationClosesAt`
- `salesEndDateTime` cannot be after `registrationClosesAt`
- Neither date can be after `eventEndDateTime`

Standard Response Format

All API responses follow a consistent structure:

Success Response Structure

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Operation completed successfully",
  "action_time": "2025-09-23T10:30:45",
  "data": {}
}
```

Error Response Structure

```
{
  "success": false,
  "httpStatus": "BAD_REQUEST",
  "message": "Error description",
  "action_time": "2025-09-23T10:30:45",
  "data": "Error description"
}
```

Standard Response Fields

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

HTTP Method Badge Standards

- **GET** — Green (`#28a745`) — Safe, read-only operations
- **POST** — Blue (`#007bff`) — Create new resources
- **PUT** — Yellow (`#ffc107`, black text) — Replace entire resource
- **PATCH** — Orange (`#fd7e14`) — Partial update
- **DELETE** — Red (`#dc3545`) — Remove resource

Enum Reference

TicketPricingType

Value	Description
`PAID`	Standard paid ticket. Price must be greater than `0.00`
`FREE`	Free entry. Price must be exactly `0.00`
`DONATION`	Supporter ticket. No fixed price — buyer freely enters their donation amount at checkout. Restricted to `ONLINE_ONLY` channel, max 1 per order and per user. `price` is `null` in responses

SalesChannel

Value	Description
`EVERYWHERE`	Available both online and at the door
`ONLINE_ONLY`	Available for purchase online only
`AT_DOOR_ONLY`	Available for purchase at the venue door only

AttendanceMode

Value	Description
`IN_PERSON`	Ticket grants physical entry to the venue
`ONLINE`	Ticket grants access to the online/virtual stream

> For `IN_PERSON` events, only `IN_PERSON` tickets are allowed. For `ONLINE` events, only `ONLINE` tickets are allowed. For `HYBRID` events, both are permitted and at least one of each is required before publishing.

TicketVisibility

Value	Description
`VISIBLE`	Always shown to the public
`HIDDEN`	Never shown to buyers (organizer use only)
`HIDDEN_WHEN_NOT_ON_SALE`	Only visible while the ticket is actively on sale
`CUSTOM_SCHEDULE`	Shown only within a defined date/time window. Requires `visibilityStartDate` and `visibilityEndDate`

TicketStatus

Value	Description
`ACTIVE`	Ticket is live and available for purchase
`INACTIVE`	Temporarily paused. Organizer can reactivate
`CLOSED`	Permanently stopped. Cannot be reopened
`SOLD_OUT`	System-managed. Set automatically when `ticketsSold >= totalTickets`. Reverts to `ACTIVE` if capacity is increased
`DELETED`	Soft-deleted. Only possible if zero tickets were sold. Use the Delete endpoint — cannot be set via status update

---

Endpoints

---

1. Create Ticket

Purpose: Creates a new ticket type for a specific event. The event must be in `DRAFT` or `PUBLISHED` status. The authenticated user must be the event organizer.

Endpoint: `POST` `{base_url}/{eventId}`

Access Level: □□ Protected (Event organizer only)

Authentication: Bearer Token

Request Headers:

Header	Type	Required	Description
`Authorization`	string	Yes	`Bearer <token>`
`Content-Type`	string	Yes	`application/json`

Path Parameters:

Parameter	Type	Required	Description	Validation
`eventId`	UUID	Yes	The event to add the ticket to	Must be a valid UUID

Request Body Parameters:

Parameter	Type	Required	Description	Validation
`name`	string	Yes	Ticket name (e.g. "VIP Pass", "Early Bird")	Min: 2, Max: 100 characters. Must be unique per event per attendance mode
`description`	string	No	Optional description of what the ticket includes	Max: 500 characters
`price`	decimal	Conditional	Ticket price. Use `0.00` for FREE tickets. Omit or send `0.00` for DONATION tickets	Min: 0.00. PAID → must be > 0.00. FREE → must be 0.00. DONATION → ignored
`ticketPricingType`	string	Yes	Pricing model	Enum: `PAID`, `FREE`, `DONATION`
`salesChannel`	string	Yes	Where the ticket can be purchased. DONATION must use `ONLINE_ONLY`	Enum: `EVERYWHERE`, `ONLINE_ONLY`, `AT_DOOR_ONLY`. Defaults to `EVERYWHERE`
`totalQuantity`	integer	Yes	Total number of tickets available	Min: 1, Max: 1,000,000
`salesStartDateTime`	datetime	No	When sales open. Must be within registration window	ISO 8601 ZonedDateTime. Cannot be before `registrationOpensAt` or after `registrationClosesAt`

Parameter	Type	Required	Description	Validation
`salesEndDateTime`	datetime	No	When sales close. Must be within registration window	ISO 8601 ZonedDateTime. Must be after `salesStartDateTime` with at least 30 minutes gap
`minQuantityPerOrder`	integer	No	Minimum tickets per order	Min: 1. Defaults to 1
`maxQuantityPerOrder`	integer	No	Maximum tickets per order. DONATION tickets must be 1	Min: 1, Max: 100. Must be ≥ `minQuantityPerOrder`
`maxQuantityPerUser`	integer	No	Maximum tickets a single user can purchase across all orders. DONATION tickets must be 1	Min: 1, Max: 1000. Must be ≥ `maxQuantityPerOrder`
`visibility`	string	Yes	Controls whether the ticket is shown to buyers	Enum: `VISIBLE`, `HIDDEN`, `HIDDEN_WHEN_NOT_ON_SALE`, `CUSTOM_SCHEDULE`. Defaults to `VISIBLE`
`visibilityStartDate`	datetime	No	When the ticket becomes visible. Required if `visibility` is `CUSTOM_SCHEDULE`	ISO 8601 ZonedDateTime
`visibilityEndDate`	datetime	No	When the ticket stops being visible. Required if `visibility` is `CUSTOM_SCHEDULE`	ISO 8601 ZonedDateTime. Must be after `visibilityStartDate`
`attendanceMode`	string	Yes	Whether this ticket is for physical or online attendance	Enum: `IN_PERSON`, `ONLINE`. Must match the event format
`inclusiveItems`	array of strings	No	List of perks included with this ticket (e.g. "Free T-Shirt", "Meet & Greet")	Max: 50 items. Each item: max 200 characters, cannot be blank

Request JSON Sample (PAID):

```
{
  "name": "VIP Pass",
  "description": "Full weekend access with backstage entry and a complimentary gift bag.",
  "price": 150.00,
  "ticketPricingType": "PAID",
```

```
  "salesChannel": "EVERYWHERE",
  "totalQuantity": 200,
  "salesStartDateTime": "2026-03-18T08:00:00+03:00",
  "salesEndDateTime": "2026-04-17T23:59:00+03:00",
  "minQuantityPerOrder": 1,
  "maxQuantityPerOrder": 4,
  "maxQuantityPerUser": 4,
  "visibility": "VISIBLE",
  "attendanceMode": "IN_PERSON",
  "inclusiveItems": [
    "Backstage access",
    "Complimentary gift bag",
    "Priority seating"
  ]
}
```

Request JSON Sample (DONATION):

```
{
  "name": "Support the Artist",
  "description": "Show your support — donate any amount you choose at checkout.",
  "price": 0.00,
  "ticketPricingType": "DONATION",
  "salesChannel": "ONLINE_ONLY",
  "totalQuantity": 500,
  "salesStartDateTime": "2026-03-18T08:00:00+03:00",
  "salesEndDateTime": "2026-04-17T23:59:00+03:00",
  "minQuantityPerOrder": 1,
  "maxQuantityPerOrder": 1,
  "maxQuantityPerUser": 1,
  "visibility": "VISIBLE",
  "attendanceMode": "IN_PERSON"
}
```

Success Response JSON Sample:

```
{
  "success": true,
  "httpStatus": "CREATED",
  "message": "Ticket created successfully",
```

```
  "action_time": "2025-02-18T10:30:45",
  "data": {
    "id": "a1b2c3d4-e5f6-7890-abcd-ef1234567890",
    "eventId": "f1e2d3c4-b5a6-7890-fedc-ba9876543210",
    "name": "VIP Pass",
    "description": "Full weekend access with backstage entry and a complimentary gift bag.",
    "price": 150.00,
    "ticketPricingType": "PAID",
    "salesChannel": "EVERYWHERE",
    "totalTickets": 200,
    "ticketsSold": 0,
    "ticketsRemaining": 200,
    "ticketsAvailable": 200,
    "isSoldOut": false,
    "salesStartDateTime": "2026-03-18T05:00:00Z",
    "salesEndDateTime": "2026-04-17T20:59:00Z",
    "isOnSale": false,
    "minQuantityPerOrder": 1,
    "maxQuantityPerOrder": 4,
    "maxQuantityPerUser": 4,
    "visibility": "VISIBLE",
    "visibilityStartDate": null,
    "visibilityEndDate": null,
    "isCurrentlyVisible": true,
    "attendanceMode": "IN_PERSON",
    "inclusiveItems": [
      "Backstage access",
      "Complimentary gift bag",
      "Priority seating"
    ],
    "status": "ACTIVE",
    "saleStatusMessage": "On sale until Apr 17, 2026",
    "createdAt": "2025-02-18T10:30:45+03:00",
    "updatedAt": null,
    "createdBy": "john_organizer",
    "updatedBy": null
  }
}
```

> **Note on DONATION response**: For `DONATION` tickets, the `price` field is `null` in the response. The buyer enters their own amount at checkout.

Success Response Fields:

Field	Description
`id`	Unique identifier for this ticket type
`eventId`	The event this ticket belongs to
`name`	Ticket name
`description`	Ticket description
`price`	Ticket price. `null` for DONATION tickets
`ticketPricingType`	Pricing model: `PAID`, `FREE`, or `DONATION`
`salesChannel`	Where the ticket can be purchased
`totalTickets`	Total number of tickets created
`ticketsSold`	Number of tickets purchased so far
`ticketsRemaining`	`totalTickets - ticketsSold`
`ticketsAvailable`	Same as `ticketsRemaining`
`isSoldOut`	`true` if `ticketsSold >= totalTickets`
`salesStartDateTime`	When ticket sales open
`salesEndDateTime`	When ticket sales close
`isOnSale`	`true` if ticket is ACTIVE and currently within the sales window
`saleStatusMessage`	Human-readable message describing the current sale state (e.g. `"On sale until Apr 17, 2026"`, `"Sales start Mar 18, 2026"`, `"Sales ended"`, `"Sold out"`)
`minQuantityPerOrder`	Minimum per order
`maxQuantityPerOrder`	Maximum per order (`null` = no limit)
`maxQuantityPerUser`	Maximum per user across all orders (`null` = no limit)
`visibility`	Visibility setting
`visibilityStartDate`	Start of custom visibility window
`visibilityEndDate`	End of custom visibility window
`isCurrentlyVisible`	Whether the ticket is currently visible to buyers
`attendanceMode`	`IN_PERSON` or `ONLINE`
`inclusiveItems`	List of perks included with the ticket

Field	Description
`status`	Ticket status: `ACTIVE`, `INACTIVE`, `CLOSED`, `SOLD_OUT`, `DELETED`
`createdAt`	Timestamp when the ticket was created
`updatedAt`	Timestamp of last update (`null` if never updated)
`createdBy`	Username of the organizer who created the ticket
`updatedBy`	Username of last person who updated the ticket

Possible Error Responses:

Status	Scenario
`401`	No or expired token
`403`	Authenticated user is not the event organizer
`404`	Event not found
`400`	Event is not in DRAFT or PUBLISHED status
`400`	Ticket name already exists for this event and attendance mode
`422`	Validation errors (missing required fields, invalid price, sales window outside registration window, gap less than 30 minutes, etc.)

Error Response Examples:

Event in invalid status (400):

```
{
  "success": false,
  "httpStatus": "BAD_REQUEST",
  "message": "Tickets can only be created for DRAFT or PUBLISHED events. Current status:
CANCELLED",
  "action_time": "2025-02-18T10:30:45",
  "data": "Tickets can only be created for DRAFT or PUBLISHED events. Current status:
CANCELLED"
}
```

Duplicate Ticket Name (400):

```
{
  "success": false,
  "httpStatus": "BAD_REQUEST",
```

```
  "message": "A ticket with name 'VIP Pass' and attendance mode 'IN_PERSON' already exists for this event",
  "action_time": "2025-02-18T10:30:45",
  "data": "A ticket with name 'VIP Pass' and attendance mode 'IN_PERSON' already exists for this event"
}
```

---

2. Update Ticket

Purpose: Updates the full details of an existing ticket type. The event must still be in `DRAFT` status. All fields are optional — only the fields you send will be updated.

Endpoint: `PUT` `{base_url}/{ticketId}`

Access Level: □□ Protected (Event organizer only)

Authentication: Bearer Token

Request Headers:

Header	Type	Required	Description
`Authorization`	string	Yes	`Bearer <token>`
`Content-Type`	string	Yes	`application/json`

Path Parameters:

Parameter	Type	Required	Description	Validation
`ticketId`	UUID	Yes	The ticket to update	Must be a valid UUID

Request Body Parameters:

Parameter	Type	Required	Description	Validation
`name`	string	No	Updated ticket name	Min: 2, Max: 100 characters. Must remain unique per event per attendance mode
`description`	string	No	Updated description	Max: 500 characters
`price`	decimal	No	Updated price. Ignored for DONATION tickets	Min: 0.00. Must be consistent with `ticketPricingType`

Parameter	Type	Required	Description	Validation
`ticketPricingType`	string	No	Updated pricing model	Enum: `PAID`, `FREE`, `DONATION`
`salesChannel`	string	No	Updated sales channel. DONATION tickets must be `ONLINE_ONLY`	Enum: `EVERYWHERE`, `ONLINE_ONLY`, `AT_DOOR_ONLY`
`totalQuantity`	integer	No	Updated total capacity	Min: 1, Max: 1,000,000
`salesStartDateTime`	datetime	No	Updated sales open time	ISO 8601 ZonedDateTime. Cannot be before `registrationOpensAt`
`salesEndDateTime`	datetime	No	Updated sales close time	ISO 8601 ZonedDateTime. At least 30 minutes after `salesStartDateTime`
`minQuantityPerOrder`	integer	No	Updated minimum per order	Min: 1
`maxQuantityPerOrder`	integer	No	Updated maximum per order. DONATION must be 1	Min: 1, Max: 100
`maxQuantityPerUser`	integer	No	Updated maximum per user. DONATION must be 1	Min: 1, Max: 1000
`attendanceMode`	string	No	Updated attendance mode	Enum: `IN_PERSON`, `ONLINE`. Must match event format rules
`inclusiveItems`	array of strings	No	Full replacement list of inclusive perks	Max: 50 items. Each: max 200 characters, cannot be blank
`visibility`	string	No	Updated visibility	Enum: `VISIBLE`, `HIDDEN`, `HIDDEN_WHEN_NOT_ON_SALE`, `CUSTOM_SCHEDULE`
`visibilityStartDate`	datetime	No	Updated visibility start	Required if changing to `CUSTOM_SCHEDULE`
`visibilityEndDate`	datetime	No	Updated visibility end	Required if changing to `CUSTOM_SCHEDULE`. Must be after start

Request JSON Sample:

```
{
  "name": "VIP Weekend Pass",
```

```
  "price": 175.00,
  "maxQuantityPerOrder": 2,
  "inclusiveItems": [
    "Backstage access",
    "Complimentary gift bag",
    "Priority seating",
    "Artist meet & greet"
  ]
}
```

Success Response JSON Sample:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Ticket updated successfully",
  "action_time": "2025-02-18T11:00:00",
  "data": {
    "id": "a1b2c3d4-e5f6-7890-abcd-ef1234567890",
    "name": "VIP Weekend Pass",
    "price": 175.00,
    "maxQuantityPerOrder": 2,
    "status": "ACTIVE",
    "updatedAt": "2025-02-18T11:00:00+03:00",
    "updatedBy": "john_organizer"
  }
}
```

Success Response Fields: Same as Create Ticket response fields.

Possible Error Responses:

Status	Scenario
`401`	No or expired token
`403`	Not the event organizer
`404`	Ticket not found
`400`	Event is not in DRAFT status
`400`	Updated name conflicts with an existing ticket on the same event

Status	Scenario
`422`	Validation errors (invalid price, quantity inconsistencies, sales window violations, etc.)

---

3. Get All Tickets for Event

Purpose: Retrieves a lightweight summary list of all active (non-deleted) tickets for a given event, ordered by creation date ascending.

Endpoint: `GET` `{base_url}/{eventId}`

Access Level: □□ Public

Authentication: None required

Path Parameters:

Parameter	Type	Required	Description	Validation
`eventId`	UUID	Yes	The event to retrieve tickets for	Must be a valid UUID

Success Response JSON Sample:


```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Tickets retrieved successfully",
  "action_time": "2025-02-18T10:30:45",
  "data": [
    {
      "id": "a1b2c3d4-e5f6-7890-abcd-ef1234567890",
      "name": "General Admission",
      "price": 25.00,
      "ticketPricingType": "PAID",
      "salesChannel": "EVERYWHERE",
      "visibility": "VISIBLE",
      "totalTickets": 1000,
      "ticketsSold": 342,
      "ticketsAvailable": 658,
      "isSoldOut": false,
```

```
      "attendanceMode": "IN_PERSON",
      "status": "ACTIVE",
      "isOnSale": true
    },
    {
      "id": "b2c3d4e5-f6a7-8901-bcde-f12345678901",
      "name": "VIP Pass",
      "price": 150.00,
      "ticketPricingType": "PAID",
      "salesChannel": "EVERYWHERE",
      "visibility": "VISIBLE",
      "totalTickets": 200,
      "ticketsSold": 200,
      "ticketsAvailable": 0,
      "isSoldOut": true,
      "attendanceMode": "IN_PERSON",
      "status": "SOLD_OUT",
      "isOnSale": false
    }
  ]
}
```


Success Response Fields (per item):

Field	Description
`id`	Unique identifier for the ticket type
`name`	Ticket name
`price`	Ticket price. `null` for DONATION tickets
`ticketPricingType`	Pricing model
`salesChannel`	Where it can be purchased
`visibility`	Visibility setting
`totalTickets`	Total quantity created
`ticketsSold`	Quantity sold so far
`ticketsAvailable`	Quantity still available
`isSoldOut`	Whether the ticket is sold out
`attendanceMode`	`IN_PERSON` or `ONLINE`
`status`	Current ticket status

Field	Description
`isOnSale`	Whether the ticket is currently purchasable
`saleStatusMessage`	Human-readable message describing the current sale state

Possible Error Responses:

Status	Scenario
`404`	Event not found

---

4. Get Ticket by ID

Purpose: Retrieves the full details of a single ticket type by its ID.

Endpoint: `GET` `{base_url}/{eventId}/{ticketId}`

Access Level: □□ Public

Authentication: None required

Path Parameters:

Parameter	Type	Required	Description	Validation
`eventId`	UUID	Yes	The event the ticket belongs to	Must be a valid UUID
`ticketId`	UUID	Yes	The specific ticket to retrieve	Must be a valid UUID

Success Response JSON Sample:


```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Ticket retrieved successfully",
  "action_time": "2025-02-18T10:30:45",
  "data": {
    "id": "a1b2c3d4-e5f6-7890-abcd-ef1234567890",
    "eventId": "f1e2d3c4-b5a6-7890-fedc-ba9876543210",
    "name": "VIP Pass",
    "description": "Full weekend access with backstage entry and a complimentary gift bag.",
```

```
    "price": 150.00,
    "ticketPricingType": "PAID",
    "salesChannel": "EVERYWHERE",
    "totalTickets": 200,
    "ticketsSold": 45,
    "ticketsRemaining": 155,
    "ticketsAvailable": 155,
    "isSoldOut": false,
    "salesStartDateTime": "2026-03-18T05:00:00Z",
    "salesEndDateTime": "2026-04-17T20:59:00Z",
    "isOnSale": true,
    "saleStatusMessage": "On sale until Apr 17, 2026",
    "minQuantityPerOrder": 1,
    "maxQuantityPerOrder": 4,
    "maxQuantityPerUser": 4,
    "visibility": "VISIBLE",
    "visibilityStartDate": null,
    "visibilityEndDate": null,
    "isCurrentlyVisible": true,
    "attendanceMode": "IN_PERSON",
    "inclusiveItems": [
      "Backstage access",
      "Complimentary gift bag",
      "Priority seating"
    ],
    "status": "ACTIVE",
    "createdAt": "2025-02-18T10:30:45+03:00",
    "updatedAt": "2025-02-18T11:00:00+03:00",
    "createdBy": "john_organizer",
    "updatedBy": "john_organizer"
  }
}
```


Success Response Fields: Same as Create Ticket response fields.

Possible Error Responses:

Status	Scenario
`404`	Event not found or ticket not found

---

5. Update Ticket Capacity

Purpose: Updates the total quantity (capacity) of a ticket. Allowed on both DRAFT and PUBLISHED events. The new capacity must be greater than or equal to the number of tickets already sold.

Endpoint: `PATCH` `{base_url}/{eventId}/{ticketId}/capacity`

Access Level: □□ Protected (Event organizer only)

Authentication: Bearer Token

Request Headers:

Header	Type	Required	Description
`Authorization`	string	Yes	`Bearer <token>`
`Content-Type`	string	Yes	`application/json`

Path Parameters:

Parameter	Type	Required	Description	Validation
`eventId`	UUID	Yes	The event	Must be a valid UUID
`ticketId`	UUID	Yes	The ticket to update capacity for	Must be a valid UUID

Request Body Parameters:

Parameter	Type	Required	Description	Validation
`newTotalQuantity`	integer	Yes	The new total capacity	Min: 1, Max: 1,000,000. Must be ≥ `ticketsSold`

Request JSON Sample:

```
{
  "newTotalQuantity": 300
}
```

Success Response JSON Sample:

```
{
  "success": true,
  "httpStatus": "OK",
```

```
  "message": "Ticket capacity updated successfully",
  "action_time": "2025-02-18T12:00:00",
  "data": {
    "id": "a1b2c3d4-e5f6-7890-abcd-ef1234567890",
    "name": "VIP Pass",
    "totalTickets": 300,
    "ticketsSold": 200,
    "ticketsRemaining": 100,
    "ticketsAvailable": 100,
    "isSoldOut": false,
    "status": "ACTIVE",
    "updatedAt": "2025-02-18T12:00:00+03:00",
    "updatedBy": "john_organizer"
  }
}
```


> **Note:** If a ticket was previously `SOLD_OUT` and the new capacity exceeds tickets sold, the status is automatically reset to `ACTIVE`.

Possible Error Responses:

Status	Scenario
`401`	No or expired token
`403`	Not the event organizer
`404`	Ticket not found
`400`	New capacity is less than the number of tickets already sold
`422`	`newTotalQuantity` is missing or below minimum

Capacity Below Sold Count (400):


```
{
  "success": false,
  "httpStatus": "BAD_REQUEST",
  "message": "Cannot reduce capacity to 100 because 200 tickets have already been sold",
  "action_time": "2025-02-18T12:00:00",
  "data": "Cannot reduce capacity to 100 because 200 tickets have already been sold"
}
```

6. Update Ticket Status

Purpose: Manually changes the status of a ticket type. Use this to pause sales (`INACTIVE`), permanently stop sales (`CLOSED`), or reactivate a paused ticket (`ACTIVE`). Works on both DRAFT and PUBLISHED events. The system automatically manages `SOLD_OUT` status — it cannot be set manually.

Endpoint: `PATCH` `{base_url}/{eventId}/{ticketId}/status`

Access Level: □□ Protected (Event organizer only)

Authentication: Bearer Token

Request Headers:

Header	Type	Required	Description
`Authorization`	string	Yes	`Bearer <token>`
`Content-Type`	string	Yes	`application/json`

Path Parameters:

Parameter	Type	Required	Description	Validation
`eventId`	UUID	Yes	The event	Must be a valid UUID
`ticketId`	UUID	Yes	The ticket to update status for	Must be a valid UUID

Request Body Parameters:

Parameter	Type	Required	Description	Validation
`status`	string	Yes	The new status to set	Enum: `ACTIVE`, `INACTIVE`, `CLOSED`. Cannot set `SOLD_OUT` or `DELETED` manually

Status Transition Rules:

Current Status	Allowed Transitions	Notes
`ACTIVE`	`INACTIVE`, `CLOSED`	Normal operations
`INACTIVE`	`ACTIVE`, `CLOSED`	Can be reactivated
`SOLD_OUT`	`ACTIVE`, `CLOSED`	`ACTIVE` only if capacity was increased first

Current Status	Allowed Transitions	Notes
`CLOSED`	None	Permanent. Cannot be changed
`DELETED`	None	Permanent. Cannot be changed

Request JSON Sample:

```
{
  "status": "INACTIVE"
}
```

Success Response JSON Sample:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Ticket status updated successfully",
  "action_time": "2025-02-18T13:00:00",
  "data": {
    "id": "a1b2c3d4-e5f6-7890-abcd-ef1234567890",
    "name": "VIP Pass",
    "status": "INACTIVE",
    "isOnSale": false,
    "updatedAt": "2025-02-18T13:00:00+03:00",
    "updatedBy": "john_organizer"
  }
}
```

Possible Error Responses:

Status	Scenario
`401`	No or expired token
`403`	Not the event organizer
`404`	Ticket not found
`400`	Attempted to set `SOLD_OUT` or `DELETED` manually
`400`	Attempted to change status of a `CLOSED` or `DELETED` ticket
`422`	`status` field is missing

7. Delete Ticket

Purpose: Soft-deletes a ticket type. The ticket is marked as deleted and hidden from all listings. **Deletion is only allowed if zero tickets have been sold.** If tickets have already been sold, close the ticket using the status endpoint instead.

Endpoint: `DELETE` `{base_url}/{eventId}/{ticketId}`

Access Level: □□ Protected (Event organizer only)

Authentication: Bearer Token

Request Headers:

Header	Type	Required	Description
`Authorization`	string	Yes	`Bearer <token>`

Path Parameters:

Parameter	Type	Required	Description	Validation
`eventId`	UUID	Yes	The event	Must be a valid UUID
`ticketId`	UUID	Yes	The ticket to delete	Must be a valid UUID

Success Response JSON Sample:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Ticket deleted successfully",
  "action_time": "2025-02-18T14:00:00",
  "data": null
}
```

Possible Error Responses:

Status	Scenario
`401`	No or expired token
`403`	Not the event organizer
`404`	Ticket not found
`400`	Ticket cannot be deleted because tickets have already been sold

Tickets Already Sold (400):

```
{
  "success": false,
  "httpStatus": "BAD_REQUEST",
  "message": "Cannot delete ticket 'VIP Pass' because 45 tickets have been sold. You can close the ticket instead to stop sales.",
  "action_time": "2025-02-18T14:00:00",
  "data": "Cannot delete ticket 'VIP Pass' because 45 tickets have been sold. You can close the ticket instead to stop sales."
}
```

---

8. Update Sales Window

Purpose: Updates the sales start and/or end datetime of a ticket on a PUBLISHED event. Both fields are optional — omitting one preserves its current value. All existing sales period rules apply (30-minute minimum gap, must fall within registration window, etc.).

Endpoint: `PATCH` `{base_url}/{ticketId}/sales-window`

Access Level: □□ Protected (Event organizer only)

Authentication: Bearer Token

Request Headers:

Header	Type	Required	Description
`Authorization`	string	Yes	`Bearer <token>`
`Content-Type`	string	Yes	`application/json`

Path Parameters:

Parameter	Type	Required	Description	Validation
`ticketId`	UUID	Yes	The ticket to update the sales window for	Must be a valid UUID

Request Body Parameters:

Parameter	Type	Required	Description	Validation

`salesStartDateTime`	datetime	No	New sales open time	ISO 8601 ZonedDateTime. Must be on or after `registrationOpensAt`. Cannot be after `registrationClosesAt`
`salesEndDateTime`	datetime	No	New sales close time	ISO 8601 ZonedDateTime. Must be after `salesStartDateTime` with at least 30 minutes gap. Cannot be after `registrationClosesAt`

> **At least one of `salesStartDateTime` or `salesEndDateTime` must be provided.**

Request JSON Sample:

```
{
  "salesEndDateTime": "2026-04-25T23:59:00+03:00"
}
```

Success Response JSON Sample:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Ticket sales window updated successfully",
  "action_time": "2026-02-20T10:00:00",
  "data": {
    "id": "a1b2c3d4-e5f6-7890-abcd-ef1234567890",
    "eventId": "f1e2d3c4-b5a6-7890-fedc-ba9876543210",
    "name": "VIP Pass",
    "salesStartDateTime": "2026-03-18T05:00:00Z",
    "salesEndDateTime": "2026-04-25T20:59:00Z",
    "isOnSale": true,
    "status": "ACTIVE",
    "updatedAt": "2026-02-20T10:00:00+03:00",
    "updatedBy": "john_organizer"
  }
}
```

Possible Error Responses:

Status	Scenario
`401`	No or expired token
`403`	Not the event organizer
`404`	Ticket not found
`400`	Ticket is deleted or closed
`422`	Sales window violates registration window, gap less than 30 minutes, or no fields provided

Sales Window Outside Registration Window (422):

```
{
  "success": false,
  "httpStatus": "UNPROCESSABLE_ENTITY",
  "message": "Sales end date cannot be after registration closes (2026-04-17T20:59:00Z)",
  "action_time": "2026-02-20T10:00:00",
  "data": {
    "stage": "TICKETS",
    "message": "Sales end date cannot be after registration closes (2026-04-17T20:59:00Z)"
  }
}
```

---

9. Update Published Ticket

Purpose: Performs a limited update on a ticket belonging to a PUBLISHED event. Only three fields are allowed: `visibility` (and its schedule dates), `status` (ACTIVE, INACTIVE, or CLOSED), and `inclusiveItems`. All other fields must be updated while the event is still in DRAFT.

Endpoint: `PATCH` `{base_url}/{ticketId}/published`

Access Level: □□ Protected (Event organizer only)

Authentication: Bearer Token

Request Headers:

Header	Type	Required	Description
`Authorization`	string	Yes	`Bearer <token>`

Header	Type	Required	Description
`Content-Type`	string	Yes	`application/json`

Path Parameters:

Parameter	Type	Required	Description	Validation
`ticketId`	UUID	Yes	The published ticket to update	Must be a valid UUID

Request Body Parameters:

Parameter	Type	Required	Description	Validation
`visibility`	string	No	Updated visibility setting	Enum: `VISIBLE`, `HIDDEN`, `HIDDEN_WHEN_NOT_ON_SALE`, `CUSTOM_SCHEDULE`
`visibilityStartDate`	datetime	No	Start of custom visibility window	Required if `visibility` is `CUSTOM_SCHEDULE`. ISO 8601 ZonedDateTime
`visibilityEndDate`	datetime	No	End of custom visibility window	Required if `visibility` is `CUSTOM_SCHEDULE`. Must be after `visibilityStartDate`. ISO 8601 ZonedDateTime
`status`	string	No	Updated ticket status	Enum: `ACTIVE`, `INACTIVE`, `CLOSED`. Cannot set `SOLD_OUT` or `DELETED`
`inclusiveItems`	array of strings	No	Full replacement list of perks	Max: 50 items. Each: max 200 characters, cannot be blank

> **This endpoint is specifically for PUBLISHED events.** For DRAFT events, use the full `PUT /{ticketId}` endpoint instead.

Request JSON Sample:

```
{
  "visibility": "CUSTOM_SCHEDULE",
  "visibilityStartDate": "2026-03-01T00:00:00+03:00",
  "visibilityEndDate": "2026-04-17T23:59:00+03:00",
```

```
  "inclusiveItems": [
    "Backstage access",
    "Complimentary gift bag",
    "Priority seating",
    "Artist meet & greet",
    "Exclusive after-party entry"
  ]
}
```

Success Response JSON Sample:


```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Ticket updated successfully",
  "action_time": "2026-02-20T10:00:00",
  "data": {
    "id": "a1b2c3d4-e5f6-7890-abcd-ef1234567890",
    "eventId": "f1e2d3c4-b5a6-7890-fedc-ba9876543210",
    "name": "VIP Pass",
    "description": "Full weekend access with backstage entry and a complimentary gift bag.",
    "price": 150.00,
    "ticketPricingType": "PAID",
    "salesChannel": "EVERYWHERE",
    "totalTickets": 200,
    "ticketsSold": 87,
    "ticketsRemaining": 113,
    "ticketsAvailable": 113,
    "isSoldOut": false,
    "salesStartDateTime": "2026-03-18T05:00:00Z",
    "salesEndDateTime": "2026-04-17T20:59:00Z",
    "isOnSale": true,
    "minQuantityPerOrder": 1,
    "maxQuantityPerOrder": 4,
    "maxQuantityPerUser": 4,
    "visibility": "CUSTOM_SCHEDULE",
    "visibilityStartDate": "2026-03-01T00:00:00+03:00",
    "visibilityEndDate": "2026-04-17T23:59:00+03:00",
    "isCurrentlyVisible": false,
    "attendanceMode": "IN_PERSON",
```

```
    "inclusiveItems": [
      "Backstage access",
      "Complimentary gift bag",
      "Priority seating",
      "Artist meet & greet",
      "Exclusive after-party entry"
    ],
    "status": "ACTIVE",
    "createdAt": "2025-02-18T10:30:45+03:00",
    "updatedAt": "2026-02-20T10:00:00+03:00",
    "createdBy": "john_organizer",
    "updatedBy": "john_organizer"
  }
}
```


Possible Error Responses:

Status	Scenario
`401`	No or expired token
`403`	Not the event organizer
`404`	Ticket not found
`400`	Event is not in PUBLISHED status — use the draft update endpoint instead
`400`	Ticket is deleted
`422`	Invalid visibility schedule dates, attempted to set `SOLD_OUT` or `DELETED`, or `CUSTOM_SCHEDULE` missing required date fields

Event Not Published (400):


```
{
  "success": false,
  "httpStatus": "BAD_REQUEST",
  "message": "This endpoint is only for published events. Use the draft ticket update endpoint instead.",
  "action_time": "2026-02-20T10:00:00",
  "data": "This endpoint is only for published events. Use the draft ticket update endpoint instead."
}
```

Quick Reference

Endpoint Summary

#	Method	Path	Description	Auth	Event Status
1	POST	`/{eventId}`	Create a ticket type	Organizer	DRAFT or PUBLISHED
2	PUT	`/{ticketId}`	Full update of ticket details	Organizer	DRAFT only
3	GET	`/{eventId}`	Get all tickets for an event	Public	Any
4	GET	`/{eventId}/{ticketId}`	Get a single ticket by ID	Public	Any
5	PATCH	`/{eventId}/{ticketId}/capacity`	Update ticket capacity	Organizer	DRAFT or PUBLISHED
6	PATCH	`/{eventId}/{ticketId}/status`	Update ticket status	Organizer	DRAFT or PUBLISHED
7	DELETE	`/{eventId}/{ticketId}`	Delete a ticket	Organizer	DRAFT or PUBLISHED (0 sold)
8	PATCH	`/{ticketId}/sales-window`	Update sales window dates	Organizer	PUBLISHED
9	PATCH	`/{ticketId}/published`	Limited update (visibility, status, perks)	Organizer	PUBLISHED only

Common HTTP Status Codes

Code	Meaning
`200 OK`	Successful GET, PATCH, PUT, or DELETE
`201 Created`	Successful POST (ticket created)
`400 Bad Request`	Business rule violated (wrong status, already sold, duplicate name)
`401 Unauthorized`	Missing or invalid token
`403 Forbidden`	Authenticated but not the event organizer
`404 Not Found`	Event or ticket does not exist
`422 Unprocessable Entity`	Validation errors on request fields

Business Rule Cheat Sheet

Rule	Detail
Create ticket	Allowed on DRAFT and PUBLISHED events
Full update (`PUT`)	DRAFT events only
Published update (`PATCH /published`)	PUBLISHED events only. Visibility, status, and inclusiveItems only
Sales window update	PUBLISHED events. Validates against registration window
Capacity update	DRAFT and PUBLISHED events
Status update	DRAFT and PUBLISHED events
FREE ticket	Price must be exactly `0.00`
PAID ticket	Price must be greater than `0.00`
DONATION ticket	`ONLINE_ONLY` channel. Max 1 per order and per user. No fixed price — buyer sets amount at checkout. `price` is `null` in response
Sales window	Must fall within the event's registration window
Sales period gap	Minimum 30 minutes between `salesStartDateTime` and `salesEndDateTime`
HYBRID event	Must have ≥ 1 `IN_PERSON` ticket and ≥ 1 `ONLINE` ticket to publish
Delete	Only allowed if `ticketsSold = 0`
SOLD_OUT	System-managed. Cannot be set manually. Auto-cleared when capacity is increased
CLOSED	Permanent. Cannot be reversed

Revision #7
Created 11 December 2025 09:55:34 by Admin Qbit
Updated 20 February 2026 09:35:18 by Admin Qbit