Event Check-In System 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 Check-In System API provides secure ticket validation and scanner management for event entry. This API enables organizers to generate registration tokens (like WhatsApp's "Link Device"), scanners to register for events, and perform real-time ticket validation with JWT signature verification. The system supports multi-day events with per-day check-in tracking, device fingerprinting for security, automatic scanner revocation, and comprehensive duplicate detection.

Hints:

- **Registration Flow**: Organizer generates token → Scanner scans QR → Scanner registers → Automatic revocation of old scanners
- **Device Security**: Fingerprint validation prevents credential theft
- **JWT Validation**: RSA-signed tickets verified offline-capable
- **Multi-Day Events**: Separate check-in per event day
- **One Device Rule**: One ACTIVE scanner per device across all events
- **Check-In Strategies**: 5 strategies (HOURS_BEFORE, SPECIFIC_TIME, ALL_DAY, EXACT_TIME, AS_DAY_START)
- **Duplicate Detection**: Prevents same-day re-entry per event day
- **Scanner Stats**: Tracks successful/failed scans automatically
- **Auto-Revocation**: Old scanners revoked when device registers for new event
- **Organizer Only**: Only event organizers can generate tokens and manage scanners

---

API Overview

Registration Token Endpoints

1. **POST** `/check-in/tokens/generate` - Generate registration token (organizer)
2. **GET** `/check-in/tokens/validate/{token}` - Validate registration token

Scanner Management Endpoints

3. **POST** `/check-in/scanners/register` - Register scanner device
4. **GET** `/check-in/scanners/event/{eventId}` - Get all scanners for event
5. **GET** `/check-in/scanners/event/{eventId}/active` - Get active scanners
6. **POST** `/check-in/scanners/{scannerId}/revoke` - Revoke scanner

Ticket Validation Endpoint

7. **POST** `/check-in/validate` - Validate ticket and check-in

Response Structures

RegistrationTokenResponse


```
{
  "tokenId": "uuid",
  "token": "REG-ABC12345-XYZ67890",
  "eventId": "uuid",
  "eventName": "East African Tech Summit 2025",
  "scannerName": "Gate A - Main Entrance",
  "expiresAt": "2025-12-11T10:35:00Z",
  "validityMinutes": 5,
  "remainingSeconds": 240,
  "qrCodeData": "scannerapp://register?token=REG-ABC12345-XYZ67890",
  "isValid": true,
  "used": false
}
```


ScannerResponse


```
{
  "scannerId": "uuid-string",
  "name": "Gate A - Main Entrance",
  "eventId": "uuid",
  "eventName": "East African Tech Summit 2025",
  "status": "ACTIVE",
  "deviceFingerprint": "abc123def456...",
  "createdAt": "2025-12-11T10:30:00Z",
```

```
  "credentials": "eyJhbGc...",
  "publicKey": "MIIBIjANBgkqhkiG9w0...",
  "revocationReason": null
}
```


ValidateTicketResponse


```
{
  "valid": true,
  "status": "VALID",
  "message": "□ Entry granted for Day 1 - Opening Day. Welcome!",
  "ticketInstanceId": "uuid",
  "ticketTypeName": "VIP Pass",
  "ticketSeries": "VIP-0001",
  "attendeeName": "John Doe",
  "attendeeEmail": "john@example.com",
  "eventName": "East African Tech Summit 2025",
  "bookingReference": "EVT-A3F4B21C",
  "alreadyCheckedIn": false,
  "previousCheckInTime": null,
  "previousCheckInLocation": null,
  "currentCheckInTime": "2025-12-15T09:15:00+03:00",
  "validationMode": "ONLINE",
  "scannerName": "Gate A - Main Entrance",
  "dayName": "Day 1 - Opening Day"
}
```

Endpoints

1. Generate Registration Token

Endpoint: POST `/check-in/tokens/generate`

Access: □□ Event Organizer Only

Request:

```
{
  "eventId": "uuid",
  "scannerName": "Gate A - Main Entrance"
}
```

Success Response: Returns RegistrationTokenResponse

Behavior:

- Validates user is event organizer
- Validates event has RSA keys (must be published)
- Generates token: `REG-{8-UUID}-{8-UUID}`
- Sets expiry: 5 minutes (configurable)
- Returns QR code data for scanner app

Errors:

- `403 FORBIDDEN`: Not event organizer
- `404 NOT_FOUND`: Event not found
- `422`: Event not published or no RSA keys

---

2. Validate Registration Token

Endpoint: GET `/check-in/tokens/validate/{token}`

Access: □□ Public (for scanner apps)

Success Response: Returns RegistrationTokenResponse with validity status

Use Case: Scanner app validates token before registration

---

3. Register Scanner

Endpoint: POST `/check-in/scanners/register`

Access: □□ Public (uses registration token)

Request:

```
{
  "registrationToken": "REG-ABC12345-XYZ67890",
  "deviceFingerprint": "abc123def456hash",
  "scannerName": "Gate A - Main Entrance",
  "deviceInfo": "{\"model\":\"iPhone 13\",\"os\":\"iOS 15\"}"
}
```

Success Response: Returns ScannerResponse with credentials

Behavior:

1. Validates device fingerprint (10-255 chars)
2. Validates token (not used, not expired)
3. **Auto-revokes** any ACTIVE scanner with same device fingerprint
4. Generates scanner ID (UUID)
5. Generates JWT credentials (1 year validity)
6. Marks token as used (one-time use)

Key Rule: One device → One ACTIVE scanner (across all events)

Errors:

- `400 BAD_REQUEST`: Invalid fingerprint, token used/expired
- `404 NOT_FOUND`: Token not found

---

4. Get Scanners for Event

Endpoint: GET `/check-in/scanners/event/{eventId}`

Access: □□ Event Organizer Only

Success Response: Returns array of ScannerResponse

Includes: All scanners (ACTIVE + REVOKED)

---

5. Get Active Scanners

Endpoint: GET `/check-in/scanners/event/{eventId}/active`

Access: □□ Event Organizer Only

Success Response: Returns array of ScannerResponse (ACTIVE only)

---

6. Revoke Scanner

Endpoint: POST `/check-in/scanners/{scannerId}/revoke?reason=Suspicious activity`

Access: □□ Event Organizer Only

Success Response: 200 OK with confirmation message

Behavior:

- Changes status to REVOKED (permanent)
- Records revocation reason and timestamp
- Scanner can no longer validate tickets

---

7. Validate Ticket and Check-In

Endpoint: POST `/check-in/validate`

Access: □□ Scanner credentials required

Request:

```
{
  "jwtToken": "eyJhbGc...",
  "scannerId": "uuid-string",
  "deviceFingerprint": "abc123def456hash",
  "checkInLocation": "Gate A"
}
```

Success Response: Returns ValidateTicketResponse

Validation Flow:

1. **Validate Scanner**: Active status, fingerprint match
2. **Verify JWT**: RSA signature with event's public key
3. **Find Current Day**: Match current time to event schedules
4. **Find Booking**: Locate ticket in database
5. **Check Duplicate**: Already checked in for this day?

6. **Create Check-In**: Add check-in record for current day
7. **Update Stats**: Increment scanner counters

Check-In Strategies:

- **HOURS_BEFORE**: X hours before event + late grace period
- **SPECIFIC_TIME**: Daily window (e.g., 08:00-23:00)
- **ALL_DAY**: Anytime on event date (00:00-23:59)
- **EXACT_TIME**: Only during event start-end
- **AS_DAY_START**: From day start (00:00) until event end + grace

Validation Statuses:

- **VALID**: □ Entry granted
- **DUPLICATE**: □ Already checked in for this day
- **INVALID_SIGNATURE**: □ JWT signature failed
- **EXPIRED**: □ Ticket validity expired
- **NOT_FOUND**: □ Ticket not in database
- **REVOKED**: □ Scanner revoked

Response Examples:

Success (VALID):

```
{
  "success": true,
  "message": "□ Entry granted for Day 1 - Opening Day. Welcome!",
  "data": {
    "valid": true,
    "status": "VALID",
    "attendeeName": "John Doe",
    "dayName": "Day 1 - Opening Day",
    "currentCheckInTime": "2025-12-15T09:15:00+03:00"
  }
}
```

Duplicate Check-In:

```
{
  "success": false,
  "message": "□ Ticket already used for Day 1 - Opening Day. Entry denied.",
  "data": {
    "valid": false,
    "status": "DUPLICATE",
```

```
    "alreadyCheckedIn": true,
    "previousCheckInTime": "2025-12-15T09:00:00+03:00",
    "previousCheckInLocation": "Gate A"
  }
}
```

System Flows

Registration Flow

Step 1: Organizer Generates Token

```
Organizer → POST /tokens/generate
← Returns: Token + QR code data
```

Step 2: Scanner Scans QR Code

```
Scanner App → Scans QR code
Extracts: "scannerapp://register?token=REG-..."
```

Step 3: Scanner Validates Token (Optional)

```
Scanner App → GET /tokens/validate/{token}
← Confirms: Token valid, event details
```

Step 4: Scanner Registers

```
Scanner App → POST /scanners/register
Sends: Token, device fingerprint, name
← Receives: Scanner credentials (JWT)
```

Step 5: Scanner Stores Credentials

```
Scanner App → Saves: credentials, publicKey
Ready to validate tickets offline
```

Ticket Validation Flow

Step 1: Scanner Scans Ticket QR

```
Scanner → Reads JWT from QR code
```

Step 2: Offline Validation (Optional)

```
Scanner → Verifies JWT signature with public key
Checks: Expiry, event match
```

Step 3: Online Check-In

```
Scanner → POST /validate
Sends: JWT, scannerId, fingerprint
```

Step 4: System Validates

```
System → Validates scanner status
System → Verifies JWT signature
System → Finds current event day
System → Checks duplicate
System → Records check-in
```

Step 5: Response to Scanner

```
System → Returns validation result
Scanner → Shows success/error to staff
```

Multi-Day Event Support

How It Works

Single-Day Event:

- One check-in expected
- Status: ACTIVE → USED after check-in

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

- Multiple check-ins allowed (one per day)
- Each day tracked separately

- Ticket stays ACTIVE throughout

Example: 3-Day Festival

JWT contains schedules:

```
{
  "eventSchedules": [
    {
      "dayName": "Day 1 - Friday Night",
      "startDateTime": "2025-12-15T18:00:00+03:00",
      "endDateTime": "2025-12-15T23:59:00+03:00"
    },
    {
      "dayName": "Day 2 - Saturday",
      "startDateTime": "2025-12-16T10:00:00+03:00",
      "endDateTime": "2025-12-16T23:59:00+03:00"
    },
    {
      "dayName": "Day 3 - Sunday",
      "startDateTime": "2025-12-17T10:00:00+03:00",
      "endDateTime": "2025-12-17T20:00:00+03:00"
    }
  ]
}
```

Check-In Timeline:

```
Friday 18:30 → Check-in for "Day 1 - Friday Night" □
Friday 19:00 → Duplicate for "Day 1" □
Saturday 11:00 → Check-in for "Day 2 - Saturday" □
Saturday 15:00 → Duplicate for "Day 2" □
Sunday 12:00 → Check-in for "Day 3 - Sunday" □
```

Validation Logic:

1. System finds current time: Saturday 11:00
2. Matches to event schedule: "Day 2 - Saturday"
3. Checks if ticket already checked in for "Day 2"
4. If no → Allow check-in
5. If yes → Reject as duplicate

Check-In Window Strategies

1. HOURS_BEFORE (Default)

Configuration:

- earlyCheckInHours: 2 (default)
- lateCheckInMinutes: 30 (default)

Window: 2 hours before event until 30 minutes after event ends

Example:

```
Event: 09:00 - 18:00
Check-in allowed: 07:00 - 18:30
```

Use Case: Conferences, concerts

2. SPECIFIC_TIME

Configuration:

- checkInOpensAt: "08:00"
- checkInClosesAt: "23:00"

Window: Same time window each event day

Example:

```
3-day event (Dec 15-17)
Check-in allowed: 08:00-23:00 each day
```

Use Case: Multi-day festivals with consistent entry hours

3. ALL_DAY

Configuration: None needed

Window: Entire event date (00:00 - 23:59)

Example:

```
Event date: Dec 15
Check-in allowed: Dec 15 00:00 - Dec 15 23:59
```

Use Case: All-day events, exhibitions

4. EXACT_TIME

Configuration: None

Window: Only during event start-end times

Example:

```
Event: 14:00 - 17:00
Check-in allowed: 14:00 - 17:00 only
```

Use Case: Strict timing events

5. AS_DAY_START

Configuration:

- lateCheckInMinutes: 30 (default)

Window: From start of event day (00:00) until event end + grace

Example:

```
Event: Dec 15 18:00 - 23:00
Check-in allowed: Dec 15 00:00 - 23:30
```

Use Case: Evening events with early arrival

Security Features

Device Fingerprinting

Purpose: Prevent credential theft

How It Works:

1. Scanner generates fingerprint from device hardware
2. Fingerprint sent with every request
3. System validates fingerprint matches registered device
4. Mismatch → Reject (credentials stolen)

Fingerprint Components (example):

```
const fingerprint = SHA256(
  deviceModel +
  osVersion +
  hardwareId +
  appInstallId
);
```

JWT Credentials

Scanner Credentials (1 year validity):

```
{
  "scannerId": "uuid",
  "eventId": "uuid",
  "type": "scanner_credential",
  "iat": 1702300000,
  "exp": 1733836000
}
```

Signed with: Event's RSA private key

Used for: Scanner authentication to API

Ticket JWT Verification

Process:

1. Scanner receives ticket JWT (QR code)
2. Scanner verifies signature with event's public key
3. Scanner can validate offline (no internet needed)
4. Scanner sends to API for check-in recording

Benefits:

- Offline validation capability

- Cannot forge tickets
- Cannot reuse tokens (duplicate tracking)
- Tamper-proof (signature validation)

Auto-Revocation System

The Rule

One device → One ACTIVE scanner at a time (across ALL events)

Scenarios

Scenario 1: Device Registers for Different Event

```
Device ABC has ACTIVE scanner for Event 1
Device ABC registers for Event 2
→ System revokes Event 1 scanner automatically
→ Creates new scanner for Event 2
```

Scenario 2: Same Event Re-Registration

```
Device ABC has ACTIVE scanner for Event 1
Device ABC registers again for Event 1
→ System revokes old scanner
→ Creates new scanner (new credentials)
```

Revocation Message:

```
"Automatically revoked: Device registered as new scanner for event 'East African Tech Summit 2025'"
```

Why This Rule?

Prevents:

- Device scanning tickets for multiple events simultaneously
- Confusion about which event device is working
- Credential misuse across events

Allows:

- Device switching between events (auto-handled)
- Fresh start for each event
- Clean scanner sessions

Scanner Statistics

Auto-Tracked Metrics

- **totalScans**: All scan attempts
- **successfulScans**: Valid entries
- **failedScans**: Duplicates, invalid tickets
- **lastScanAt**: Most recent scan timestamp
- **lastSyncedAt**: Last API contact

Success Rate Calculation

```
successRate = (successfulScans / totalScans) * 100
```

Updated Automatically

- Each validation attempt updates counters
- Successful → successfulScans++, totalScans++
- Failed → failedScans++, totalScans++

Error Codes Summary

Registration Token Errors

- **400**: Token expired/used, invalid format
- **403**: Not event organizer
- **404**: Token/event not found
- **422**: Event not published, no RSA keys

Scanner Registration Errors

- **400**: Invalid fingerprint (too short/long)
- **400**: Token expired/used
- **404**: Token not found
- **422**: Invalid scanner name

Ticket Validation Errors

- **INVALID_SIGNATURE**: JWT signature failed
- **DUPLICATE**: Already checked in for this day
- **EXPIRED**: Ticket validity expired
- **NOT_FOUND**: Ticket not in database
- **REVOKED**: Scanner revoked

Best Practices

For Organizers

- [ ] Generate tokens right before scanner setup
- [ ] Use descriptive scanner names (e.g., "Gate A - Main")
- [ ] Monitor scanner activity via dashboard
- [ ] Revoke suspicious scanners immediately
- [ ] Test scanner before event starts

For Scanner App Developers

- [ ] Generate stable device fingerprint
- [ ] Store credentials securely (encrypted)
- [ ] Implement offline validation first
- [ ] Sync check-ins when online
- [ ] Show clear success/error messages
- [ ] Handle device fingerprint mismatch gracefully

For Event Staff

- [ ] Keep scanners charged
- [ ] Verify scanner name matches gate
- [ ] Watch for duplicate warnings
- [ ] Report technical issues immediately
- [ ] Use backup manual verification if needed

Quick Reference

Token Lifetime

- Registration token: **5 minutes**
- Scanner credentials: **1 year**
- Ticket JWT: **Event validity period**

Device Fingerprint

- Min length: **10 characters**
- Max length: **255 characters**
- Format: **Stable hash of device properties**

Scanner Name

- Min length: **3 characters**
- Max length: **200 characters**
- Examples: "Gate A", "VIP Entrance", "Main Hall Scanner"

Status Flow

```
Registration Token: unused → used (one-time)
Scanner: ACTIVE → REVOKED (permanent)
Validation: VALID | DUPLICATE | INVALID_SIGNATURE | EXPIRED | NOT_FOUND | REVOKED
```

Integration Guide

Organizer Dashboard Integration

```
// 1. Generate token
POST /check-in/tokens/generate
{
  eventId: "uuid",
```

```
  scannerName: "Gate A"
}

// 2. Display QR code
<QRCode value={response.qrCodeData} />

// 3. Show token expiry countdown
remainingTime = response.remainingSeconds

// 4. List active scanners
GET /check-in/scanners/event/{eventId}/active
```

Scanner App Integration

```
// 1. Scan registration QR code
const token = extractTokenFromQR(qrData);

// 2. Register device
POST /check-in/scanners/register
{
  registrationToken: token,
  deviceFingerprint: generateFingerprint(),
  scannerName: "Gate A",
  deviceInfo: JSON.stringify(deviceDetails)
}

// 3. Store credentials
secureStorage.save('credentials', response.credentials);
secureStorage.save('publicKey', response.publicKey);
secureStorage.save('scannerId', response.scannerId);

// 4. Validate tickets
while (eventActive) {
  const ticketJWT = scanTicketQR();

  // Offline validation
  const offlineValid = verifyJWT(ticketJWT, publicKey);

  if (offlineValid) {
```

```
    // Online check-in
    POST /check-in/validate
    {
      jwtToken: ticketJWT,
      scannerId: scannerId,
      deviceFingerprint: generateFingerprint(),
      checkInLocation: "Gate A"
    }
  }
}
```

Conclusion

The Event Check-In System provides enterprise-grade security with:

- **WhatsApp-Style Registration**: Scan QR to link device
- **Device Security**: Fingerprint validation prevents theft
- **Offline Capable**: JWT verification without internet
- **Multi-Day Support**: Per-day check-in tracking
- **Auto-Revocation**: Clean scanner sessions per event
- **Flexible Timing**: 5 check-in window strategies
- **Duplicate Prevention**: Same-day re-entry blocked
- **Real-Time Stats**: Automatic success/fail tracking

Revision #1
Created 11 December 2025 10:16:18 by Admin Qbit
Updated 11 December 2025 10:16:58 by Admin Qbit