

Payout Management System Documentation

Table of Contents

- [Overview](#)
- [System Architecture](#)
- [Payment Flow](#)
- [Database Schema](#)
- [API Endpoints](#)
- [User Interfaces](#)
- [Webhook Integration](#)
- [Admin Instructions](#)
- [Configuration](#)
- [Testing](#)

Overview

The Callastar Payout Management System provides a complete solution for managing creator payments, from booking payment to final payout. The system implements a **7-day holding period** for security and dispute management, followed by a **manual admin approval process** for payout requests.

Key Features

- **✓ Automatic Payment Hold:** All payments are held for 7 days after booking
- **✓ Creator Payout Requests:** Creators can request payouts for available balance
- **✓ Admin Approval Workflow:** Admins manually process payout requests
- **✓ Stripe Integration:** Direct transfers to creator Stripe Connect accounts
- **✓ Webhook Automation:** Automatic status updates from Stripe
- **✓ Analytics Dashboard:** Comprehensive platform metrics and insights

System Architecture

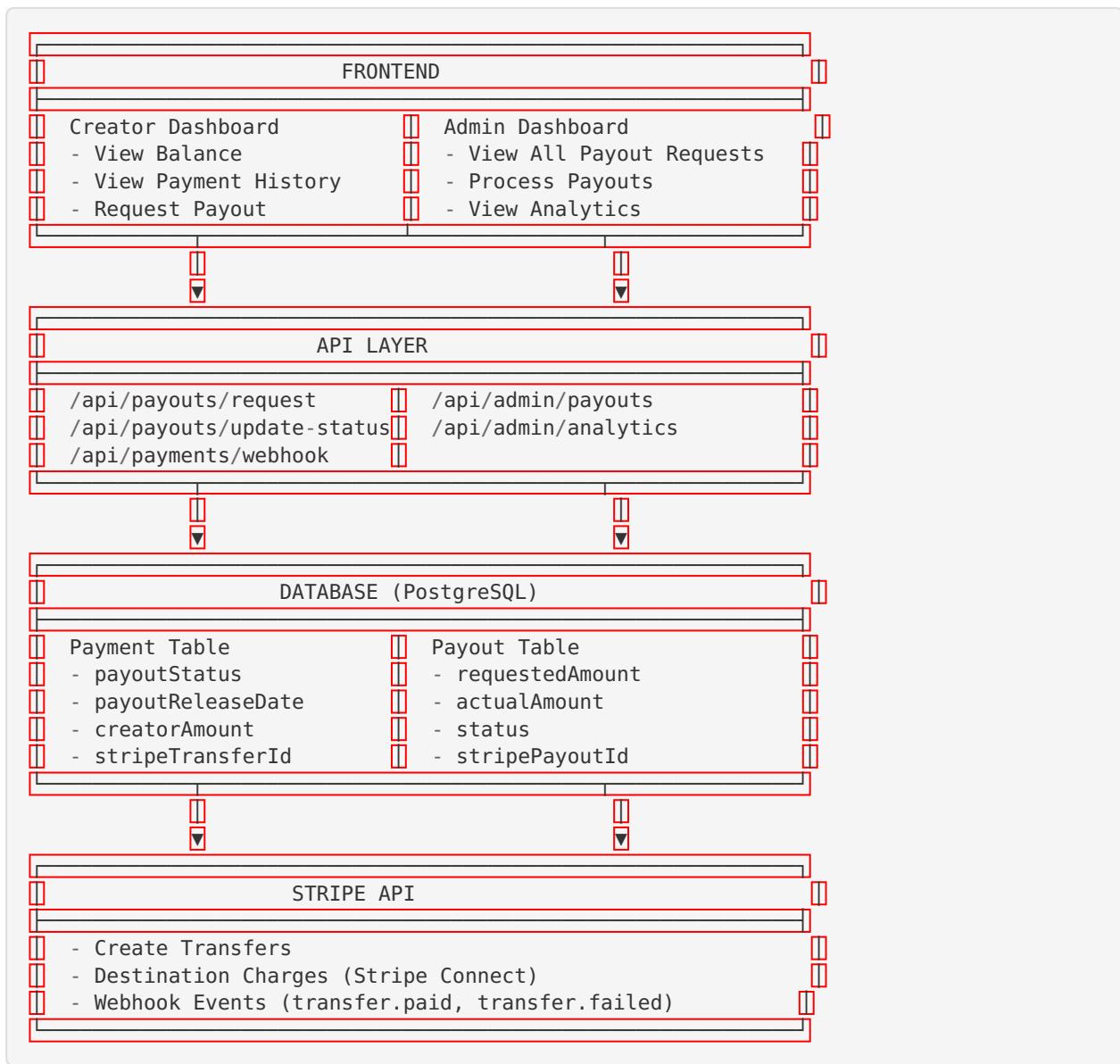
Payment Status Flow

```

BOOKING PAYMENT
↓
PAYMENT CREATED (status: SUCCEEDED)
↓
HELD (7 days security period)
↓ (automatic after 7 days via cron job)
READY (available for payout)
↓ (creator requests payout)
PAYOUT REQUEST CREATED (status: PENDING)
↓ (admin processes request)
PROCESSING (Stripe transfer initiated)
↓ (Stripe webhook confirms)
PAID (funds transferred to creator)

```

Component Architecture



\$ Payment Flow

1. Initial Booking Payment

When a user books a call:

1. Payment Intent Created:

- Uses Stripe Connect Destination Charges
- 10% platform fee automatically deducted
- 90% designated for creator

2. Payment Record Created (via webhook):

```
typescript
{
  status: 'SUCCEEDED',
  payoutStatus: 'HELD',
  payoutReleaseDate: currentDate + 7 days,
  amount: 70.00,           // Total amount in EUR
  platformFee: 7.00,       // 10% platform fee
  creatorAmount: 63.00    // 90% for creator
}
```

2. Holding Period (7 Days)

- Payments remain in `HELD` status for 7 days
 - Protects against chargebacks and disputes
 - **Cron Job** runs daily via `/api/payouts/update-status` :
- ```
bash
Checks if payoutReleaseDate has passed
Updates payoutStatus from HELD → READY
```

## 3. Creator Payout Request

When release date is reached:

1. Creator sees available balance in dashboard
2. Creator clicks “Request Payout” button
3. Enters amount to withdraw (up to available balance)
4. System creates **Payout Request**:

```
typescript
{
 status: 'PENDING',
 requestedAmount: 63.00,
 stripeAccountId: 'acct_xxx',
 requestedAt: new Date()
}
```

## 4. Admin Processing

Admin reviews and processes requests:

1. Views pending requests in `/dashboard/admin/payouts`
2. Verifies creator’s Stripe account status
3. Clicks “Process” button

4. System:

- Creates Stripe Transfer to creator account
- Updates payout status to `PROCESSING`
- Updates payment records to `PROCESSING`

## 5. Stripe Transfer Completion

Stripe webhook handles completion:

1. `transfer.paid event received`:

- Updates payout status to `PAID`
- Updates payment records to `PAID`
- Sends notification to creator
- Sends email confirmation

2. `transfer.failed event received`:

- Updates payout status to `FAILED`
- Reverts payment records to `READY`
- Notifies creator of failure

## Database Schema

### Payment Model

```
model Payment {
 id String @id @default(cuid())
 bookingId String @unique
 amount Decimal @db.Decimal(10, 2)
 stripePaymentIntentId String
 status PaymentStatus @default(PENDING)
 platformFee Decimal @db.Decimal(10, 2)
 creatorAmount Decimal @db.Decimal(10, 2)
 createdAt DateTime @default(now())

 // Payout tracking
 payoutStatus PayoutStatus @default(HELD)
 payoutReleaseDate DateTime?
 stripeTransferId String?
 payoutDate DateTime?

 booking Booking @relation(fields: [bookingId], references: [id])
 @@index([payoutStatus])
}
```

## Payout Model

```
model Payout []
 id String @id @default(cuid())
 creatorId String
 requestedAmount Decimal @db.Decimal(10, 2)
 actualAmount Decimal? @db.Decimal(10, 2)
 stripePayoutId String?
 stripeAccountId String?
 status PayoutStatus @default(PENDING)

 // Tracking dates
 requestedAt DateTime @default(now())
 processedAt DateTime?
 completedAt DateTime?
 failedAt DateTime?

 // Metadata
 failureReason String? @db.Text
 notes String? @db.Text
 processedBy String?

 createdAt DateTime @default(now())
 updatedAt DateTime @updatedAt

 creator Creator @relation(fields: [creatorId], references: [id])
 @@index([creatorId])
 @@index([status])
}
```

## PayoutStatus Enum

```
enum PayoutStatus {
 PENDING // Initial state, awaiting payment
 HELD // Payment succeeded, funds held for 7 days
 READY // Holding period passed, ready for transfer
 PROCESSING // Transfer in progress
 PAID // Successfully transferred to creator
 FAILED // Transfer failed
 CANCELLED // Cancelled (refund/dispute)
}
```

## API Endpoints

### Creator APIs

**GET /api/payouts/request**

Get creator's balance and payment details.

**Response:**

```
{
 "availableBalance": 126.00,
 "pendingBalance": 63.00,
 "totalPaid": 315.00,
 "stripeConnected": true,
 "payments": {
 "ready": [...],
 "held": [...],
 "paid": [...]
 },
 "pendingPayoutRequests": [...]
}
```

### POST /api/payouts/request

Create a payout request.

#### Request:

```
{
 "amount": 126.00
}
```

#### Response:

```
{
 "success": true,
 "message": "Demande de paiement de 126.00€ créée avec succès",
 "payout": {
 "id": "payout_xxx",
 "amount": 126.00,
 "status": "PENDING"
 }
}
```

## Admin APIs

### GET /api/admin/payouts

List all payout requests with filters.

#### Query Parameters:

- status : Filter by status (PENDING, PROCESSING, PAID, FAILED)
- creatorId : Filter by creator ID

#### Response:

```
[
 {
 "id": "payout_xxx",
 "creatorName": "John Doe",
 "creatorEmail": "john@example.com",
 "stripeAccountId": "acct_xxx",
 "requestedAmount": 126.00,
 "availableBalance": 126.00,
 "status": "PENDING",
 "requestedAt": "2025-12-25T10:00:00Z"
 }
]
```

### POST /api/admin/payouts

Process a payout request.

#### Request:

```
{
 "payoutId": "payout_xxx",
 "notes": "Processed manually"
}
```

#### Response:

```
{
 "success": true,
 "message": "Paiement de 126.00€ transféré avec succès",
 "payout": {
 "id": "payout_xxx",
 "amount": 126.00,
 "stripePayoutId": "tr_xxx",
 "status": "PAID"
 }
}
```

### GET /api/admin/analytics

Get platform analytics and metrics.

#### Query Parameters:

- `period` : Number of days (default: 30)

#### Response:

```
{
 "revenue": {
 "total": 7000.00,
 "period": 700.00
 },
 "platformFees": {
 "total": 700.00,
 "period": 70.00
 },
 "payouts": {
 "pending": { "count": 5, "amount": 315.00 },
 "completed": { "count": 20, "amount": 1260.00 }
 },
 "topCreators": [...]
}
```

## Cron Job API

**POST /api/payouts/update-status**

Update payment status from HELD to READY (runs daily).

**Headers:**

```
Authorization: Bearer <CRON_SECRET>
```

**Response:**

```
{
 "message": "Updated 5 payments to READY status",
 "updatedCount": 5,
 "paymentIds": ["pay_1", "pay_2", ...]
}
```



## User Interfaces

### Creator Dashboard ( /dashboard/creator/payouts )

Features:

- **Balance Cards:** Available, Pending, Total Paid
- **Request Payout Button:** Creates payout request
- **Payment Lists:**
  - Ready payments (can be included in payout)
  - Held payments (with countdown)
  - Paid history
- **Payout Request Status:** Shows pending requests

### Admin Payout Management ( /dashboard/admin/payouts )

Features:

- **Summary Cards:** Pending, Processing, Completed amounts
- **Payout Requests Table:**
- Creator details

- Requested vs available balance
- Request date
- Status badge
- Action buttons
- **Process Button:** Initiates Stripe transfer
- **Stripe Dashboard Link:** Opens creator's Stripe account
- **Status Filtering:** Filter by status

## Admin Analytics ( /dashboard/admin/analytics )

Features:

- **Period Selector:** 7, 30, 90, 365 days
- **Revenue Metrics:**
  - Total revenue
  - Platform fees
  - Creator earnings
- **Payout Metrics:**
  - Pending amount
  - Completed payouts
  - Current balances
- **Charts:**
  - Revenue over time
  - Payment status distribution
- **Top Creators:** Highest earners in period

## 🔔 Webhook Integration

### Stripe Webhook Configuration

#### 1. Configure Webhook Endpoint:

```
https://your-domain.com/api/payments/webhook
```

#### 2. Required Events:

- `payment_intent.succeeded` - Initial payment capture
- `transfer.paid` - Payout completed successfully
- `transfer.failed` - Payout failed

#### 3. Webhook Secret:

```
bash
Add to .env
STRIPE_WEBHOOK_SECRET=whsec_xxxxx
```

## Webhook Handler ( /api/payments/webhook/route.ts )

Handles three event types:

#### 1. `payment_intent.succeeded`

- Creates Payment record
- Sets payoutStatus to HELD
- Calculates payoutReleaseDate (7 days)
- Sends confirmation emails

## 2. `transfer.paid`

- Updates Payout status to PAID
- Updates Payment records to PAID
- Sends success notification to creator
- Records transfer ID and date

## 3. `transfer.failed`

- Updates Payout status to FAILED
- Reverts Payment records to READY
- Records failure reason
- Sends error notification to creator



# Admin Instructions

## Daily Operations

### 1. Check Pending Payout Requests

Navigate to `/dashboard/admin/payouts` :

- View all pending requests
- Check creator's available balance
- Verify Stripe account status

### 2. Process Payout Request

#### Steps:

1. Click "Stripe" button to verify creator's account:

- Check if payouts are enabled
- Verify bank account is connected
- Review account status

1. Click "Process" button:

- System creates Stripe transfer
- Status updates to PROCESSING
- Wait for webhook confirmation

2. Verify Completion:

- Check payout status changes to PAID
- Verify in Stripe Dashboard
- Creator receives email notification

### 3. Handle Failed Payouts

If payout fails:

1. Check failure reason in admin panel
2. Contact creator to resolve issue:
  - Bank account issues
  - Stripe account verification needed
  - Insufficient funds (rare)
3. Creator can request again after fixing

## 4. Monitor Analytics

Check `/dashboard/admin/analytics` regularly:

- **Revenue Trends:** Monitor platform growth
- **Pending Payouts:** Ensure timely processing
- **Balance Health:** Check held vs ready amounts
- **Top Creators:** Identify high performers

## Troubleshooting

### Creator Can't Request Payout

#### Possible Causes:

1. Stripe Connect not configured
  - Guide creator to `/dashboard/creator/settings`
  - Complete Stripe onboarding
1. No available balance
  - Payments still in HELD status
  - Check payoutReleaseDate
2. Pending request exists
  - Process existing request first
  - Only one pending request allowed

### Payout Processing Fails

#### Possible Causes:

1. Invalid Stripe account
    - Verify account ID is correct
    - Check account status in Stripe
  1. Stripe API error
    - Check Stripe API logs
    - Verify API keys are correct
  2. Insufficient balance (rare)
    - Check platform's Stripe balance
    - Contact Stripe support if needed
-

## Configuration

### Environment Variables

```
Stripe Configuration
STRIPE_SECRET_KEY=sk_test_xxxxxx
STRIPE_PUBLISHABLE_KEY=pk_test_xxxxxx
STRIPE_WEBHOOK_SECRET=whsec_xxxxxx

Cron Job Secret
CRON_SECRET=your-secure-secret

App URLs
NEXT_PUBLIC_APP_URL=https://your-domain.com
NEXTAUTH_URL=https://your-domain.com

Database
DATABASE_URL=postgresql://user:pass@host:5432/db
```

### Cron Job Setup

Set up daily cron job to update payment statuses:

**Vercel Cron** (vercel.json):

```
{
 "crons": [
 {
 "path": "/api/payouts/update-status",
 "schedule": "0 0 * * *"
 }
]
}
```

**Alternative (External Cron):**

```
Call daily at midnight
0 0 * * * curl -X POST \
-H "Authorization: Bearer YOUR_CRON_SECRET" \
https://your-domain.com/api/payouts/update-status
```

### Stripe Constants

Located in `lib/stripe.ts` :

```
export const PLATFORM_FEE_PERCENTAGE = 10; // 10% platform fee
export const PAYOUT_HOLDING_DAYS = 7; // 7 day hold period
```

# 🧪 Testing

## Test Scenarios

### 1. Complete Payment Flow

```
1. Create booking and payment
2. Wait for payment_intent.succeeded webhook
3. Verify Payment created with HELD status
4. Check payoutReleaseDate is +7 days

5. Simulate cron job (or wait 7 days)
curl -X POST \
 -H "Authorization: Bearer YOUR_CRON_SECRET" \
 http://localhost:3000/api/payouts/update-status

6. Verify payment status changed to READY
7. Login as creator and request payout
8. Login as admin and process payout
9. Wait for transfer.paid webhook
10. Verify payment marked as PAID
```

### 2. Test Payout Request Validation

```
Test: Request exceeds balance
POST /api/payouts/request
{ "amount": 999999 }
Expected: 400 error

Test: Stripe not configured
Remove stripeAccountId from creator
POST /api/payouts/request
{ "amount": 10 }
Expected: 400 error

Test: Already has pending request
Create request, then try again
POST /api/payouts/request
{ "amount": 10 }
Expected: 400 error
```

### 3. Test Webhook Events

```
Use Stripe CLI to forward webhooks
stripe listen --forward-to localhost:3000/api/payments/webhook

Trigger test events
stripe trigger payment_intent.succeeded
stripe trigger transfer.paid
stripe trigger transfer.failed
```

## Manual Testing Checklist

- [ ] Creator can view balance correctly
- [ ] Creator can request payout
- [ ] Admin sees pending requests
- [ ] Admin can view creator's Stripe account

- [ ] Admin can process payout
  - [ ] Webhook updates payout status
  - [ ] Creator receives email notification
  - [ ] Analytics show correct metrics
  - [ ] Held→Ready status update works
  - [ ] Failed payout reverts to READY
- 

## Notes

### Security Considerations

1. **Authorization:** All APIs check user role (CREATOR/ADMIN)
2. **Validation:** Request amounts validated against available balance
3. **Idempotency:** Webhook handler checks for existing payments
4. **Cron Secret:** Protected endpoint requires secret header

### Performance Optimizations

1. **Indexed Fields:** payoutStatus, status, creatorId
2. **Pagination:** Limit paid payment history to last 50
3. **Caching:** Consider caching analytics for large datasets

### Future Enhancements

- [ ] Automatic payout processing (optional)
  - [ ] Bulk payout processing
  - [ ] Custom holding periods per creator
  - [ ] CSV export of payout history
  - [ ] Email notifications for payout milestones
  - [ ] Multi-currency support
  - [ ] Scheduled payouts (weekly/monthly)
- 

## Support

For issues or questions:

1. Check [Stripe Documentation](https://stripe.com/docs/connect) (<https://stripe.com/docs/connect>)
  2. Review application logs
  3. Contact platform admin
  4. Submit GitHub issue
- 

**Last Updated:** December 25, 2025

**Version:** 1.0.0

**Author:** Callastar Development Team