Currency Management Strategy

Overview

This document explains the comprehensive currency management system implemented for payouts in the “Call a Star” platform. The system handles multi-currency Stripe Connect accounts while maintaining EUR as the platform’s base currency.

Date: December 26, 2024
Version: 1.0
Feature Branch: `feature/stripe-payout-automation`

Table of Contents

Core Strategy

EUR as Source of Truth

Principle: All amounts are stored in the database in EUR, regardless of the Stripe account currency.

Why EUR?
- ✅ **Consistent accounting:** Single currency for all financial records
- ✅ **Simple reconciliation:** Easy to track platform revenue and fees
- ✅ **Transparent reporting:** Clear financial statements
- ✅ **Historical accuracy:** Conversion rates tracked separately

Conversion at Payout Time

Principle: Currency conversion happens at the moment of payout request, not when payments are received.

Flow:
1. User earns money → Stored in EUR (database)
2. User requests payout → System checks Stripe account currency
3. If currency ≠ EUR → Convert EUR to Stripe currency

4. Create Stripe payout → Use converted amount in Stripe currency
5. Store both amounts → EUR (original) and Stripe currency (paid)

Database Schema

Payout Model Changes

```
model Payout {
  id              String       @id @default(cuid())
  creatorId       String

  // Original amount in EUR (source of truth)
  amount          Decimal      @db.Decimal(10, 2)

  // NEW: Currency tracking fields
  amountPaid      Decimal?     @db.Decimal(10, 2)  // Amount paid in Stripe currency
  currency        String?      @default("EUR")      // Stripe account currency
  conversionRate  Decimal?     @db.Decimal(10, 6)  // EUR → Stripe currency rate
  conversionDate  DateTime?                         // When conversion was done

  stripePayoutId  String?
  status          PayoutStatus @default(PENDING)
  failureReason   String?      @db.Text
  retriedCount    Int          @default(0)
  createdAt       DateTime     @default(now())
  updatedAt       DateTime     @updatedAt

  creator         Creator          @relation(fields: [creatorId], references: [id], on
Delete: Cascade)
  transactionLogs TransactionLog[] @relation("PayoutLogs")

  @@index([creatorId])
  @@index([status])
  @@index([currency])
}
```

Migration

```
-- Location: prisma/migrations/20251226120020_add_currency_tracking_to_payouts/migra-
tion.sql

ALTER TABLE "Payout" ADD COLUMN "amountPaid" DECIMAL(10,2);
ALTER TABLE "Payout" ADD COLUMN "currency" TEXT DEFAULT 'EUR';
ALTER TABLE "Payout" ADD COLUMN "conversionRate" DECIMAL(10,6);
ALTER TABLE "Payout" ADD COLUMN "conversionDate" TIMESTAMP(3);
CREATE INDEX "Payout_currency_idx" ON "Payout"("currency");
```

Implementation Details

File Structure

```
lib/
  ├── currency-converter.ts          # Currency conversion utilities
  └── stripe-account-validator.ts    # Existing Stripe validation

app/api/
  ├── stripe/balance/[creatorId]/route.ts  # Balance API with currency
  └── payouts/request/route.ts             # Payout request with conversion

app/dashboard/creator/
  ├── page.tsx                       # Revenue page (updated)
  ├── payouts/page.tsx               # Payouts page (updated)
  └── payouts/request/page.tsx       # Payout request page (updated)

prisma/
  ├── schema.prisma                  # Updated Payout model
  └── migrations/
      └── 20251226120020_add_currency_tracking_to_payouts/
          └── migration.sql
```

Key Functions

1. Get Stripe Currency

```
// lib/currency-converter.ts
export async function getStripeCurrency(stripeAccountId: string): Promise<string>
```

- Fetches the Stripe account's `default_currency`
- Returns uppercase currency code (e.g., 'EUR', 'CHF', 'USD')
- Fallback to 'EUR' if unable to fetch

2. Currency Conversion

```
// lib/currency-converter.ts
export async function convertEurToStripeCurrency(
  amountEur: number,
  targetCurrency: string
): Promise<CurrencyConversion>
```

- Converts EUR amount to target currency
- Uses fallback fixed rates (should be replaced with API)
- Returns conversion details (amount, rate, timestamp)

3. Conversion Rate Fetching

```
// lib/currency-converter.ts
export async function getConversionRate(
  fromCurrency: string,
  toCurrency: string
): Promise<number>
```

- Gets current conversion rate

- Currently uses fallback fixed rates
- **TODO:** Integrate with exchange rate API (e.g., exchangerate-api.com)

API Endpoints

Balance API

Endpoint: `GET /api/stripe/balance/[creatorId]`

Changes:
- Added `stripeCurrency` field to response
- Fetches Stripe account details to get default currency

Response:

```
{
  "available": 100.50,
  "pending": 25.00,
  "currency": "EUR",           // Database currency
  "stripeCurrency": "CHF",      // Stripe account currency
  "accountStatus": { ... }
}
```

Payout Request API

Endpoint: `POST /api/payouts/request`

Changes:
1. Get Stripe account currency
2. Check if conversion needed
3. Convert amount if necessary
4. Create payout in Stripe with converted amount
5. Store conversion details in database

Request:

```
{
  "amount": 100.00  // Amount in EUR
}
```

Response (with conversion):

```json
{
  "success": true,
  "message": "Paiement de 100.00 EUR (≈ 93.00 CHF) en cours de transfert",
  "payout": {
    "id": "pyt_xxx",
    "amountEur": 100.00,
    "amountPaid": 93.00,
    "currency": "CHF",
    "conversionRate": 0.93,
    "stripePayoutId": "po_xxx",
    "status": "processing"
  }
}
```

Flow:

```
// 1. Get Stripe currency
const stripeCurrency = await getStripeCurrency(creator.stripeAccountId);

// 2. Convert if needed
if (stripeCurrency !== 'EUR') {
  const conversion = await convertEurToStripeCurrency(amountEur, stripeCurrency);
  amountPaid = conversion.toAmount;
  conversionRate = conversion.rate;
}

// 3. Create Stripe payout
const stripePayout = await createConnectPayout({
  amount: amountPaid,
  currency: stripeCurrency.toLowerCase(),
  stripeAccountId: creator.stripeAccountId,
});

// 4. Store with conversion details
await prisma.payout.create({
  data: {
    amount: amountEur,             // Original EUR
    amountPaid: amountPaid,         // Converted amount
    currency: stripeCurrency,
    conversionRate: conversionRate,
    conversionDate: new Date(),
  },
});
```

Frontend Components

1. Revenue Page (Dashboard Creator)

File: `app/dashboard/creator/page.tsx`

Changes:
- ✅ Removed “Withdraw Money” button
- ✅ Made page strictly informative
- ✅ Added link to Payouts page
- ✅ Display amounts with EUR currency explicitly

Before:

```
<Button onClick={handleRequestPayout}>
  Demander le paiement
</Button>
```

After:

```
<Link href="/dashboard/creator/payouts">
  <Button variant="outline">
    Voir les payouts
  </Button>
</Link>
```

2. Payouts Page

File: `app/dashboard/creator/payouts/page.tsx`

Changes:
- ✅ Show Stripe currency alongside EUR
- ✅ Display conversion info when applicable
- ✅ Add currency information alert

Example:

```
<div className="text-3xl font-bold">
  {balance?.available.toFixed(2)} {balance?.stripeCurrency}
</div>
{balance?.stripeCurrency !== 'EUR' && (
  <p className="text-sm text-gray-500">
    ≈ {balance?.available.toFixed(2)} EUR (base)
  </p>
)}
```

3. Payout Request Page

File: `app/dashboard/creator/payouts/request/page.tsx`

Changes:
- ✅ Show Stripe currency in balance display
- ✅ Add conversion preview alert
- ✅ Explain conversion will happen at payout time

Currency Conversion Alert:

```
{balance?.stripeCurrency !== 'EUR' && (
  <Alert className="bg-blue-50 border-blue-200">
    <AlertDescription>
      Conversion de devise: Votre compte Stripe est en {balance.stripeCurrency}.
      Le montant sera converti automatiquement de EUR vers {balance.stripeCurrency}.
    </AlertDescription>
  </Alert>
)}
```

Currency Conversion

Current Implementation

Method: Fallback fixed rates

Rates (Approximate, as of Dec 2024):

```
const rates = {
  EUR: {
    CHF: 0.93,  // 1 EUR = 0.93 CHF
    USD: 1.10,  // 1 EUR = 1.10 USD
    GBP: 0.85,  // 1 EUR = 0.85 GBP
    CAD: 1.47,  // 1 EUR = 1.47 CAD
    AUD: 1.63,  // 1 EUR = 1.63 AUD
  }
};
```

Recommended Production Solution

Option 1: Exchange Rate API (Recommended)
- Service: exchangerate-api.com (https://www.exchangerate-api.com/)
- Free tier: 1,500 requests/month
- Live rates updated regularly

Implementation:

```
export async function getConversionRate(from: string, to: string): Promise<number> {
  const API_KEY = process.env.EXCHANGE_RATE_API_KEY;
  const response = await fetch(
    `https://v6.exchangerate-api.com/v6/${API_KEY}/pair/${from}/${to}`
  );
  const data = await response.json();
  return data.conversion_rate;
}
```

Option 2: Stripe Rates
- Use Stripe's internal conversion rates
- More consistent with Stripe's actual conversion
- Requires additional Stripe API calls

Option 3: Fixed Rates (Current)
- Simple, no external dependencies
- Must be updated regularly (weekly/monthly)
- Less accurate, but sufficient for most cases

Testing Scenarios

Test Case 1: EUR Account (No Conversion)

Setup:
- Creator has EUR Stripe account
- Available balance: 100 EUR

Expected Behavior:
1. Balance shown: “100.00 EUR”
2. No conversion alert displayed
3. Payout request: 100 EUR
4. Stripe receives: 100 EUR
5. Database stores:
- `amount` : 100.00
- `currency` : “EUR”
- `amountPaid` : NULL (or 100.00)
- `conversionRate` : NULL (or 1.0)

Success Criteria:
- ✅ No conversion happens
- ✅ Amounts match exactly
- ✅ No conversion info shown in UI

Test Case 2: CHF Account (Conversion)

Setup:
- Creator has CHF Stripe account
- Available balance: 100 EUR (database)
- Conversion rate: 1 EUR = 0.93 CHF

Expected Behavior:
1. Balance shown: “100.00 CHF” with “≈ 100.00 EUR (base)”
2. Conversion alert displayed
3. Payout request: 100 EUR
4. System converts: 100 EUR → 93 CHF
5. Stripe receives: 93 CHF
6. Database stores:
- `amount` : 100.00 (EUR)
- `currency` : “CHF”
- `amountPaid` : 93.00 (CHF)
- `conversionRate` : 0.93
- `conversionDate` : [timestamp]

Success Criteria:
- ✅ Conversion happens correctly
- ✅ Both amounts stored in database
- ✅ Conversion details visible in UI
- ✅ Correct amount sent to Stripe

Test Case 3: Insufficient Balance

Setup:
- Available balance: 5 EUR
- Minimum payout: 10 EUR

Expected Behavior:
- Error message: “Montant insuffisant”
- Cannot request payout

Test Case 4: Currency Not Compatible

Setup:
- Stripe account in unsupported currency (e.g., JPY)

Expected Behavior:
- Fallback conversion rate: 1.0 (or error)
- Warning logged
- Admin notified

Error Handling

1. Conversion API Failure

Scenario: Exchange rate API is down or rate limit exceeded

Handling:

```
try {
  rate = await getConversionRate(from, to);
} catch (error) {
  console.error('Conversion API failed:', error);
  // Fallback to fixed rates
  rate = getFallbackRates(from, to);
  // Log warning for admin review
  await logWarning('Currency conversion fallback used', { from, to, error });
}
```

2. Unsupported Currency

Scenario: Stripe account uses unsupported currency

Handling:

```
if (!isSupportedCurrency(stripeCurrency)) {
  return NextResponse.json(
    {
      error: `Devise ${stripeCurrency} non supportée. Veuillez contacter le support.`,
      supportedCurrencies: ['EUR', 'CHF', 'USD', 'GBP', 'CAD', 'AUD']
    },
    { status: 400 }
  );
}
```

3. Stripe Payout Creation Failure

Scenario: Stripe rejects payout (insufficient balance, account issues)

Handling:

```
try {
  const stripePayout = await createConnectPayout({ ... });
} catch (error) {
  // Update payout status to FAILED
  await prisma.payout.update({
    where: { id: payout.id },
    data: {
      status: 'FAILED',
      failureReason: error.message,
    },
  });

  // Return detailed error
  return NextResponse.json(
    { error: `Erreur Stripe: ${error.message}` },
    { status: 500 }
  );
}
```

Future Improvements

Phase 1: Live Exchange Rates ⭐

Priority: High
Estimated Effort: 2 hours

Tasks:
1. Sign up for exchangerate-api.com (free tier)
2. Add API key to environment variables
3. Update `getConversionRate()` function
4. Add rate caching (1 hour TTL)
5. Test with real CHF account

Benefits:
- Accurate conversion rates
- Automatic updates
- Better user experience

Phase 2: Historical Rate Tracking

Priority: Medium
Estimated Effort: 4 hours

Tasks:
1. Create `CurrencyRate` model in Prisma
2. Store daily/hourly rates
3. Use historical rates for past payouts
4. Add rate comparison charts in admin

Benefits:
- Accurate historical reporting
- Rate trend analysis
- Better accounting

Phase 3: Multi-Currency Balance Display

Priority: Low
Estimated Effort: 6 hours

Tasks:
1. Show balance in both EUR and Stripe currency
2. Add real-time conversion preview
3. Allow users to see earnings in preferred currency
4. Add currency selector in settings

Benefits:
- Better user experience
- Clearer earnings overview
- Reduced confusion

Phase 4: Dynamic Currency Switching

Priority: Low
Estimated Effort: 8 hours

Tasks:
1. Allow creators to change Stripe account currency
2. Recalculate historical balances
3. Update all payouts to show in new currency
4. Add currency change audit log

Benefits:
- Flexibility for international creators
- Support for relocating creators

Deployment Checklist

Before Deployment

- [ ] Run Prisma migration
- [ ] Test with EUR account (no conversion)
- [ ] Test with CHF account (conversion)
- [ ] Verify Stripe payout creation
- [ ] Check database records
- [ ] Test frontend currency display
- [ ] Verify error handling
- [ ] Review logs for warnings

Database Migration

```bash
# Run migration
npx prisma migrate deploy

# Verify migration
npx prisma db pull

# Check Payout table structure
npx prisma studio
```

Environment Variables

```
# Optional: Exchange rate API (for production)
EXCHANGE_RATE_API_KEY=your_api_key_here

# Existing Stripe keys remain unchanged
STRIPE_SECRET_KEY=sk_xxx
NEXT_PUBLIC_STRIPE_PUBLISHABLE_KEY=pk_xxx
```

Monitoring

Key Metrics to Track:
1. Conversion rate accuracy
2. Failed payouts due to currency issues
3. Conversion API usage (rate limits)
4. User confusion indicators (support tickets)

Logs to Monitor:
- `[Currency] Using fallback rate` - Should be rare in production
- `[Payout] Currency conversion: X EUR -> Y CHF` - Should match expectations
- Stripe API errors related to currency

Support & Troubleshooting

Common Issues

Issue 1: Amounts seem different in Stripe vs Database

Cause: Different currencies
Solution: Check `stripeCurrency` field in balance API response

Issue 2: Conversion rate seems wrong

Cause: Using fallback fixed rates
Solution: Implement live exchange rate API

Issue 3: User can’t request payout

Cause: Currency not supported or account not configured
Solution: Check `isSupportedCurrency()` and Stripe account status

Contact

For questions or issues:
- **Developer:** Check this documentation and currency-converter.ts
- **Admin:** Review payout audit logs and conversion details
- **Support:** Refer users to currency information alerts in UI

Changelog

Version 1.0 (December 26, 2024)

Added:
- ✅ Currency tracking in Payout model
- ✅ Stripe currency detection in balance API
- ✅ Currency conversion utilities
- ✅ Conversion at payout time
- ✅ Frontend currency display updates
- ✅ Conversion preview in payout request
- ✅ Comprehensive documentation

Changed:
- ✅ Removed withdraw button from Revenue page
- ✅ Made Revenue page strictly informative
- ✅ Updated all amounts to show currency explicitly

Fixed:
- ✅ 400 error when requesting payout with non-EUR currency
- ✅ Confusion about amounts in different currencies

Appendix

Supported Currencies

Currency	Code	Symbol	Supported
Euro	EUR	€	✅ Primary
Swiss Franc	CHF	CHF	✅ Yes
US Dollar	USD	$	✅ Yes
British Pound	GBP	£	✅ Yes
Canadian Dollar	CAD	CA$	✅ Yes
Australian Dollar	AUD	A$	✅ Yes

Conversion Rate Sources

Current: Fallback fixed rates (approximate)
Recommended: exchangerate-api.com (https://www.exchangerate-api.com/)
Alternative: Stripe conversion rates

Related Documentation

- STRIPE_VALIDATION_LOGIC_FIX.md (./STRIPE_VALIDATION_LOGIC_FIX.md) - Stripe account validation
- PAYOUT_FIXES_SUMMARY.md (./PAYOUT_FIXES_SUMMARY.md) - Previous payout fixes
- prisma/schema.prisma (./prisma/schema.prisma) - Database schema

---

Document End