

# Reusable Stripe Refund Function

Developer Guide · createStripeRefund(options)

## 1. Overview

The reusable **createStripeRefund(options)** function creates Stripe refunds from a backend service. It supports full and partial refunds and can be reused across E-commerce and POS modules.

**File:** src/services/payment/stripeRefund.service.js

## 2. Import

```
const {
  createStripeRefund
} = require('../../services/payment/stripeRefund.service');
```

Adjust the relative path based on the calling file's location.

## 3. Parameters

| Parameter       | Required?   | Description                                                                                             |
|-----------------|-------------|---------------------------------------------------------------------------------------------------------|
| dbName          | Conditional | Tenant database name; required unless a Stripe client is supplied.                                      |
| module          | Conditional | Credential context, such as "ecom" or "pos"; required unless a Stripe client is supplied.               |
| service         | No          | Optional additional credential context.                                                                 |
| paymentIntentId | One of two  | Stripe PaymentIntent ID (pi_...).                                                                       |
| chargeId        | One of two  | Stripe Charge ID (ch_...).                                                                              |
| amountMinor     | No          | Positive integer in the currency's smallest unit. Omit to request the full remaining refundable amount. |
| reason          | No          | requested_by_customer (default), duplicate, or fraudulent.                                              |
| idempotencyKey  | Yes         | Stable key of 1–255 characters for the same logical refund operation.                                   |
| metadata        | No          | Additional key-value information attached to the refund.                                                |
| stripeAccount   | No          | Connected Stripe account ID, when using Stripe Connect.                                                 |
| stripe          | Conditional | Existing Stripe client instance; takes precedence over dbName and module.                               |

**Key rules:** Provide exactly one payment reference (paymentIntentId or chargeId). When amountMinor is supplied, it must be a positive integer and must not exceed the remaining refundable amount. Use the original payment currency's minor-unit rules. Reuse the same idempotency key for retries of the same operation; use a different key for a separate refund.

## 4. Usage examples

### A. Full refund (E-commerce)

```
const refund = await createStripeRefund({
  dbName: "demo_store",
  module: "ecom",
  service: "ecommerce",
  paymentIntentId: "pi_example123",
  reason: "requested_by_customer",
  idempotencyKey: "cancel:demo_store:order123",
  metadata: {
    orderId: "order123",
    cancelledBy: "admin"
  }
});
```

### B. Partial refund (USD \$25.00)

```
const refund = await createStripeRefund({
  dbName: "demo_store",
```

```
module: "ecom",
service: "ecommerce",
paymentIntentId: "pi_example123",
amountMinor: 2500,
reason: "requested_by_customer",
idempotencyKey: "return:demo_store:return123",
metadata: { returnRequestId: "return123" }
});
```

### C. POS refund (USD \$15.50)

```
const refund = await createStripeRefund({
  dbName: "demo_store",
  module: "pos",
  service: "pos",
  chargeId: "ch_example123",
  amountMinor: 1550,
  idempotencyKey: "pos-return:receipt998",
  metadata: { receiptNo: "receipt998" }
});
```

## 5. Return value

The function returns the Stripe Refund object. Example:

```
{
  id: "re_example123",
  object: "refund",
  amount: 2500,
  currency: "usd",
  status: "succeeded",
  payment_intent: "pi_example123",
  metadata: { orderId: "order123" }
}
```

**Common statuses:** succeeded (refund succeeded), pending (still processing), failed (refund failed). Treat the returned status as the current refund state; a successful API response does not necessarily mean the customer's bank has completed the credit.

## 6. Validation and errors

The service should validate the payment reference, amount, idempotency key, and metadata before sending the request. Validation errors and Stripe API errors are surfaced through the application's CustomError mechanism, according to the implementation. Callers should handle errors through the project's normal error middleware.

## 7. Unit tests

Run the service unit tests from the backend project directory:

```
npx.cmd jest test/unit/stripeRefund.test.js --runInBand
```

The documented test suite covers payment reference validation, amount validation, metadata, idempotency keys, and error handling. Run the command to verify the current test results.