

The first important distinction:

etlobe dispute ≠ *Stripe dispute*.

A customer can “win a dispute” in two different ways:

1. **Internal etlobe dispute:** customer complains inside your app, etlobe/vendor decides customer deserves full or partial refund. In this case, Stripe does **not** decide the dispute. Your backend creates a Stripe **Refund**.
2. **Stripe/card dispute / chargeback:** customer goes to their bank/card issuer and disputes the payment. In this case, the bank/card network process starts. Stripe immediately debits the disputed amount and dispute fee from your Stripe balance, then you can submit evidence. If the customer wins, the money is already gone from your side. Stripe says a formal card dispute immediately reverses the payment and debits your balance for the amount and dispute fee. ([Stripe Docs](#))

For etlobe, most normal marketplace complaints should start as **internal disputes**, not card chargebacks.

1. Normal checkout flow

Assume etlobe uses **Stripe Connect with separate charges and transfers**, which is the best model for multi-vendor marketplace carts.

Stripe describes this model as useful when one transaction can involve multiple connected accounts, such as an e-commerce marketplace with one cart containing goods from multiple businesses. ([Stripe Docs](#))

The flow:

Customer adds product to cart

etlobe creates internal Order = pending_payment

etlobe creates Stripe Checkout Session / PaymentIntent

Customer pays

Stripe confirms payment

Stripe sends webhook to etlobe

etlobe marks order as paid

etlobe ledger creates vendor payable, platform commission, delivery payable, VAT/tax li

Vendor money is held internally, not paid immediately

The developer must store:

```
order_id
payment_intent_id
charge_id
stripe_customer_id if available
amount_paid
currency
vendor_id
order_item_ids
transfer_group
payment_status
refund_status
dispute_status
vendor_payout_status
```

For marketplace architecture, I would use a Stripe `transfer_group` like:

```
ORDER_123456
```

This lets your system connect the original customer payment with later vendor transfers, delivery transfers, refunds, and transfer reversals.

2. Do not transfer vendor money immediately

This is critical.

After payment, do **not** instantly pay the vendor.

Keep the vendor amount as:

```
vendor_pending_balance
```

Then release later after:

```
delivery completed
+ customer confirmation / auto-confirmation
+ no open dispute
+ no refund request
+ no chargeback
+ reserve window passed
+ finance/payout rule allows release
```

This makes refunds much easier. If the customer wins a dispute before vendor payout, you simply refund the customer and reduce the vendor's payable. You do not need to chase the vendor.

3. Customer opens an internal etlobe dispute

Example:

Customer says:

"I received the wrong item."

In your app:

```
Order status: delivered
Customer clicks: Report problem
Customer chooses: Wrong item received
Customer uploads photos
Customer requests: Full refund
```

Your backend creates:

```
DisputeCase
- dispute_id
- order_id
- order_item_id
- customer_id
- vendor_id
- reason
- requested_resolution
- requested_amount
- evidence
- status = awaiting_vendor_response
```

At this moment, **nothing happens in Stripe yet.**

Stripe does not know about this internal dispute. This is etlobe's business process.

Your backend should immediately do:

```
freeze affected vendor payable
freeze affected order item settlement
block vendor payout for this item
record dispute event in ledger/audit log
```

4. Vendor/admin decision

The dispute can end in different ways.

Case A — Customer wins full refund

Example:

```
Product price: €100
Delivery: €7
Total order line impact: €107
Decision: customer gets full refund
```

etlobe decides:

```
refund_amount = 10700 cents  
liable_party = vendor / carrier / platform / CP  
return_required = yes/no
```

Then your backend creates a Stripe refund.

Stripe's Refund API allows refunding a previous charge or PaymentIntent. The money is refunded to the original card/payment method. Stripe also says you can refund only part of a charge and do so multiple times until the full charge is refunded. ([Stripe Docs](#))

Pseudo-code:

```
const refund = await stripe.refunds.create({  
  payment_intent: "pi_xxx",  
  amount: 10700,  
  reason: "requested_by_customer",  
  metadata: {  
    order_id: "ORDER_123",  
    dispute_id: "DSP_456",  
    refund_type: "internal_marketplace_dispute",  
    liability_owner: "vendor"  
  }  
});
```

If you want a **full refund**, you can either pass the full amount or omit `amount` depending on your implementation style. Stripe says by default the entire charge is refunded, and partial refunds are created by setting an `amount`. ([Stripe Docs](#))

Case B — Customer wins partial refund

Example:

```
Product price: €100  
Customer keeps product  
Decision: €25 partial refund
```

Your backend creates:

```
const refund = await stripe.refunds.create({  
  payment_intent: "pi_xxx",  
  amount: 2500,  
  reason: "requested_by_customer",  
  metadata: {  
    order_id: "ORDER_123",  
    dispute_id: "DSP_456",  
    refund_type: "partial_refund",  
    liability_owner: "vendor"  
  }  
});
```

Stripe allows multiple partial refunds, but you cannot refund a total greater than the original charge amount. ([Stripe Docs](#))

So your system must check:

```
remaining_refundable_amount =  
amount_paid  
- previous_successful_refunds  
- pending_refunds  
- chargeback_amount_if_any
```

Never trust only the frontend amount.

5. What Stripe does during the refund

When your backend creates the refund:

Stripe receives refund request

Stripe creates Refund object

Stripe sends money back to the original payment method

Stripe deducts the refund amount from your Stripe balance

Stripe sends webhook events

etlobe updates internal refund and ledger status

Stripe says refunds go back to the credit/debit card that was originally charged. ([Stripe Docs](#))

Refunds are not always instant from the customer's perspective. Stripe says refund timing depends on the customer's bank/card issuer, and failed refunds can happen if the customer's bank or card issuer cannot process them. If a refund fails, Stripe returns the amount to your Stripe balance. ([Stripe Docs](#))

So your app should not show:

"Refund completed in your bank."

It should show:

"Refund processed. Your bank may take several days to show it."

6. What happens to the vendor money?

This depends on whether you already transferred money to the vendor.

Situation 1 — Vendor was not paid yet

Best case.

Customer paid €107
Vendor payable was pending
Customer wins €107 refund
Stripe refunds customer
etlobe reduces vendor payable to €0
No vendor transfer happens

Internal ledger:

Dr Vendor payable / vendor held funds
Cr Refund payable

Dr Refund payable
Cr Stripe clearing

Developer work:

mark refund as succeeded
reduce vendor pending balance
close dispute
mark order item refunded
prevent vendor payout
create credit note if needed

Situation 2 — Vendor already received transfer, but payout not yet gone to bank

You may be able to reverse the Stripe transfer.

Stripe says that for separate charges and transfers, refunding a charge does **not** automatically affect associated transfers. Your platform must reconcile the amount owed back by reducing later transfers or reversing transfers. ([Stripe Docs](#))

Stripe also says Connect supports full or partial transfer reversals, and transfer reversals should be used only for refunds, disputes, or transfer errors. ([Stripe Docs](#))

So after refund:

```
const reversal = await stripe.transfers.createReversal(
  "tr_vendor_transfer_id",
  {
    amount: 10700,
    metadata: {
      order_id: "ORDER_123",
      dispute_id: "DSP_456",
      reason: "customer_refund"
    }
  }
);
```

For partial refund:


```
const reversal = await stripe.transfers.createReversal(
  "tr_vendor_transfer_id",
  {
    amount: 2500
  }
);
```

Your app must not assume:

refund customer = vendor money automatically returned

That is false in the separate charges/transfers model.

Situation 3 — Vendor was already paid out to bank

This is the dangerous case.

Stripe can refund the customer from your platform balance, but recovering from the vendor may be difficult. Your system must create:

vendor_negative_balance
or vendor_receivable
or reserve_usage
or future_payout_deduction

Stripe's docs say that in Connect refund/dispute cases, the platform balance is debited, and in most cases the platform can recover by reversing the transfer. If a connected account becomes negative due to refund or dispute, Stripe can hold a reserve on its available balance. ([Stripe Docs](#))

This is why etlobe should not pay vendors too early.

7. Full refund example

Customer buys:

Phone case: €20

Delivery: €5

Total paid: €25

Vendor sends wrong item. Customer opens etlobe dispute and wins.

Before refund

Stripe payment: €25 captured

Order: paid

Vendor payable: €18

Platform commission: €2

Delivery payable: €5

Vendor payout: held

Dispute: opened

Decision

Customer gets full €25 refund

Vendor is liable for product price

Platform may absorb delivery depending policy

Stripe action

```
stripe.refunds.create({  
  payment_intent: "pi_xxx",  
  amount: 2500  
});
```

etlobe action

refund_status = pending/succeeded

order_item_status = refunded

vendor_payable -= €18

commission_revenue reversed
delivery revenue reversed or marked platform loss
customer notified
vendor notified
settlement updated

8. Partial refund example

Customer buys:

Headphones: €100
Delivery: €7
Total paid: €107

Product works, but packaging damaged. Customer keeps product. etlobe decides customer gets €20 partial refund.

Stripe action

```
stripe.refunds.create({  
  payment_intent: "pi_xxx",  
  amount: 2000  
});
```

etlobe ledger

Customer refund: €20
Vendor liability: €20 or shared
Vendor payable reduced by €20
Order remains delivered
Warranty remains active unless policy says otherwise
Invoice/credit note updated

The customer receives €20 back to the original payment method. The vendor still gets paid the remaining eligible amount later.

9. Now the other case: real Stripe/card dispute

This is different.

Customer does **not** use etlobe support. Customer goes to bank and says:

"I did not receive the item"

"This was fraud"

"Product was not as described"

Then:

Customer contacts card issuer

Issuer opens card dispute/chargeback

Stripe creates Dispute object

Payment is immediately reversed

Stripe debits platform balance for disputed amount + dispute fee

Stripe sends webhook to etlobe

etlobe must freeze vendor payout and prepare evidence

Stripe says a dispute happens when a cardholder questions the payment with their issuer, and you can respond with evidence that the charge was legitimate. ([Stripe Docs](#))

Stripe also says that at the end, if the issuer overturns the dispute in your favor, the amount is returned to Stripe and Stripe passes it back to you. If the issuer upholds the dispute in the cardholder's favor, no money moves from your perspective because Stripe already credited the issuer when the chargeback started. ([Stripe Docs](#))

So when the customer "wins" a **card dispute**, you usually do **not** create a normal refund. The chargeback already pulled the money.

Your dev team must avoid this mistake:

Card dispute opened
+ etlobe also sends refund
= double loss

10. Card dispute flow in etlobe

When Stripe sends:

`charge.dispute.created`

etlobe should:

mark `payment_dispute_status` = `chargeback_opened`
freeze vendor payout
freeze affected order items
block manual refund unless finance override
create liability case
notify admin/support
collect evidence

Evidence can include:

customer account info
order confirmation
delivery proof
tracking
CP pickup scan
customer messages
vendor proof
refund policy
photos
IP/device info for fraud cases

Stripe lets you manage disputes through API, including uploading evidence, responding to disputes, and receiving dispute events through webhooks. ([Stripe Docs](#))

If etlobe loses:

dispute_status = lost
customer already got money through card issuer
vendor liability decided internally
vendor balance reduced / negative balance created
platform absorbs dispute fee unless contract passes it to vendor

If etlobe wins:

dispute_status = won
Stripe returns disputed amount to platform balance
vendor payout can be released if no other holds

11. How full/partial refund is different from chargeback

Situation	Who starts it?	Who decides?	Stripe object	Money movement
Internal full refund	Customer in etlobe app	etlobe/vendor/admin	Refund	etlobe calls refund API
Internal partial refund	Customer in etlobe app	etlobe/vendor/admin	Refund	etlobe calls refund API with amount
Card chargeback	Customer via bank	Issuer/card network	Dispute	Stripe debits platform immediately
Customer wins card dispute	Bank/card issuer	Issuer/card network	Dispute	No extra refund; money already reversed
etlobe wins card dispute	Bank/card issuer	Issuer/card network	Dispute	Stripe passes money back to etlobe

12. Developer responsibilities

Stripe will not build your marketplace logic. Stripe manages the payment objects, but your devs must build the business brain.

Devs must build before payment

```
create Order
create OrderItems
calculate vendor splits
calculate delivery split
calculate platform commission
calculate tax/VAT fields
create PaymentIntent/Checkout Session
attach metadata
set transfer_group
```

Important metadata:

```
order_id
customer_id
cart_id
vendor_ids
environment
marketplace_country
```

Devs must build after payment

Listen to webhooks:

```
payment_intent.succeeded
charge.succeeded
checkout.session.completed
```

Then:

- mark order paid
- create ledger entries
- create vendor pending payable
- create delivery payable
- do not payout yet

Devs must build internal dispute system

- DisputeCase
- DisputeMessage
- DisputeEvidence
- DisputeDecision
- LiabilityCase
- RefundDecision

Stripe does not decide whether the product was wrong, damaged, late, fake, or missing. etlobe does.

Devs must build refund calculation

Before calling Stripe:

- check original payment exists
- check refund is allowed
- check no active chargeback
- check total previous refunds
- check remaining refundable amount
- check who is liable
- check whether vendor payout already happened
- check whether transfer reversal is needed

Devs must call Stripe Refund API

Full refund:


```
await stripe.refunds.create({
  payment_intent: paymentIntentId,
  metadata: {
    order_id,
    dispute_id
  }
});
```

Partial refund:

```
await stripe.refunds.create({
  payment_intent: paymentIntentId,
  amount: 2500,
  metadata: {
    order_id,
    dispute_id
  }
});
```

Devs must handle refund webhooks

Stripe has events for refunds, including `refund.created` , `refund.updated` , and `charge.refunded` ; `charge.refunded` includes partial refunds. ([Stripe Docs](#))

Your backend should listen and update:

```
refund_status
order_refund_status
ledger_status
customer notification
vendor settlement
invoice/credit note status
```

Devs must reverse vendor transfers when needed

If vendor transfer already happened:

```
await stripe.transfers.createReversal(transferId, {  
  amount: amountToRecover  
});
```

If transfer did not happen:

do not reverse anything
just reduce vendor payable internally

Devs must handle card dispute webhooks

Listen to:

charge.dispute.created
charge.dispute.updated
charge.dispute.closed

Then:

freeze settlement
block duplicate refund
create chargeback case
prepare evidence
decide vendor liability
recover through reserve/future payout/negative balance

Stripe recommends webhooks for dispute-created events on Connect platforms, after which platforms can try to recover funds by reversing transfers. ([Stripe Docs](#))

13. The safest etlobe rule

For your marketplace, I would implement this rule:

Internal dispute open = freeze vendor payout
Internal refund approved = call Stripe refund API
Stripe chargeback open = block manual refund

Vendor transfer not yet made = reduce vendor payable
Vendor transfer already made = reverse transfer if possible
Vendor already paid out = create negative balance / recover from reserve / deduct futur

This protects etlobe from double loss.

14. The exact backend state machine I recommend

Payment states

requires_payment
payment_pending
paid
payment_failed
partially_refunded
fully_refunded
chargeback_opened
chargeback_lost
chargeback_won

Refund states

none
requested
approved
stripe_refund_pending
stripe_refund_succeeded
stripe_refund_failed
cancelled

Vendor settlement states

not_created
pending_delivery
held_dispute
held_refund
available
transfer_created
transfer_reversed
paid_out
negative_balance

Dispute states

opened
awaiting_vendor
awaiting_customer
platform_review
customer_won_full
customer_won_partial
vendor_won
refund_processing
closed

15. What I would tell the devs

Build this as three separate systems connected by IDs:

Stripe Payment System

- PaymentIntent
- Charge
- Refund
- Transfer
- TransferReversal
- Dispute

etlobe Dispute System

- customer complaint
- vendor response
- admin decision
- full/partial refund decision
- liability owner

etlobe Ledger/Settlement System

- vendor payable
- platform commission
- refund payable
- vendor reserve
- transfer recovery
- negative balance

Never let one Stripe webhook directly “decide the marketplace truth.” Stripe tells you money movement. etlobe decides business liability.

The most important engineering sentence:

Refunding the customer and recovering money from the vendor are two separate operations. Stripe can refund the customer, but etlobe must decide whether to reduce vendor payable, reverse a transfer, use a reserve, or create a vendor negative balance.