indition™

# Subscription Products — Operations Runbook

Diagnosing and fixing things in production.

Subscription Products Documentation · August 4, 2026

## First checks, in order

1. Is **Subscriptions Enabled** on? Everything no-ops when it is off.
2. Are the two scheduled jobs present and active (Admin → Scheduled Jobs)?
3. Did the renewal job run? Check its execution log.
4. Read the subscription's **event history** — it records every decision.

## Nothing is billing

| CAUSE | CHECK | FIX |
|---|---|---|
| Module off | Settings | Switch on |
| Job missing or inactive | Admin → Scheduled Jobs → "Subscriptions: Process Renewals" | Re-run the install migrations, or add it |
| Cron not running at all | Other modules' jobs also stalled? | Host-level |
| Nothing actually due | Grid → Due in 7 Days | Not a fault |
| All paused | Status filter | Resume |

Run it by hand to see the per-subscription output:

```
DOMAIN=<site-domain> php yiic jobRunner
```

## Renewals create orders but take no money

Expected when **Auto-Charge Renewals** is off — orders are raised pending for staff to collect. That is the default.

With auto-charge on and money still not moving, check the subscription's `payment_profile_id` and the history for the gateway message. Common causes:

- No stored profile — the originating order paid with a one-off token, or the

gateway did not vault it.

- Card expired.
- Gateway unsupported: only Stripe and Authorize.Net can charge off-session.
- SCA required — comes back as a decline and enters dunning; the customer must

re-authorise.

## A customer was charged twice

Rare by design, but the paths that could cause it:

1. **Manual Renew Now racing the cron.** Mitigated by a per-subscription

advisory lock; a losing caller is reported as *skipped*.

1. **A hard crash mid-run.** Documented limitation: the cron runner wraps the

whole run in a transaction, so an order created in that run is not yet committed when the card is charged. If the process is killed (OOM, fatal), the order rolls back while the charge stands. The Stripe idempotency key and the paid-order detection bound this to that crash window; closing it fully needs per-subscription commits in the cron runner itself.

To confirm: compare `PaymentTransaction` rows against renewal orders for that subscription. A charge with no surviving order is this case.

## A renewal order has no line items

The engine detects this and fails the renewal into dunning rather than leaving an unusable order — `saveOrderDetails` can swallow per-line failures and still return the order. Usually a missing `delivery_method_id`. Check the product type's delivery methods.

## Dunning is not behaving

Retry limit resolution, in order: the plan's own **Retry limit** if set; the site default (**Max card retries before pausing**) if the plan's is blank; `0` means retry forever.

First retry waits **grace days** if configured, later retries use **days between retries**. When retries are exhausted the subscription is **paused** with no resume date, so the auto-resume sweep cannot restart billing on a card that has not been fixed.

**Resuming clears the failure count.** Without that a dunning-paused subscription would come back with its counter already past the limit and re-pause on the very next hiccup.

## The subscription paused itself

Two ways: a customer/staff pause, or dunning exhaustion. Tell them apart from the history — an automatic one says "Paused automatically after N failed payment attempts". A timed pause resumes on its date; a dunning pause never does.

## Emails are not arriving

1. Is that specific email switched on? Each of the five is independent.
2. **Is the body populated?** Renewals and reminders run under cron, and the CMS

template pipeline needs a live web request, so CLI sends always use the **body** field. A template selected with a blank body sends nothing. This is the most common cause.

1. Does the customer have an email? The current account address is used, falling

back to the originating order's.

1. Check the platform mail log.

Reminders are idempotent per (subscription, renewal date, days-before) — a marker in the event trail suppresses re-sends. If one did not arrive, look for its `renewal_reminder` event: present means it was attempted.

## A refund cancelled a subscription unexpectedly

Only a refund that makes the order **fully** refunded cancels its subscriptions. Partial refunds — one line of a mixed order, a shipping adjustment, goodwill — deliberately leave the agreement alone. If a full refund was not intended, the subscription can be restarted from the admin detail screen.

## Renewal totals look wrong

Renewal carts do **not** go through the live-checkout cart processor, whose availability processors are built for live checkouts and would drop items. They are priced through the same shipping and tax authorities a live checkout uses, so a renewal a year later is taxed at the current jurisdiction rate, not one frozen from the first order.

If tax looks wrong, check `SHOP:TAX_COMPONENT`. A related trap: with that setting **absent**, VertexTax's hook gate used to default to "Vertex configured" while the app defaulted to the standard tax component — so an unconfigured site called an unreachable SOAP endpoint on every order. Fixed in VertexTax; set the setting explicitly regardless.

## Recovering a stuck subscription

- **Wrong next date** — Edit Terms.
- **Should not have been cancelled** — Restart from the detail screen.
- **Completed too early** — raise Max Cycles under Edit Terms, then Restart.
- **Stuck past due** — fix the card, then Renew Now.
- **Duplicate subscriptions from one order line** — the database's unique partial

index prevents this; if you see it, capture the rows before deleting.

## Useful queries

```
-- Due now
SELECT id, customer_id, sub_product_id, next_renewal_date, status, failed_attempts
FROM   <site>_shop_subscription
WHERE  status IN ('active','trial','past_due') AND next_renewal_date <= CURRENT_DATE
ORDER  BY next_renewal_date;

-- Recent failures with the gateway message
SELECT s.id, e.creation_datetime, e.note
FROM   <site>_shop_subscription_event e
JOIN   <site>_shop_subscription s ON s.id = e.subscription_id
WHERE  e.event_type = 'renewal_failed'
ORDER  BY e.creation_datetime DESC LIMIT 50;

-- Live agreements with no payment profile (will fail under auto-charge)
SELECT id, customer_id, sub_product_id
FROM   <site>_shop_subscription
WHERE  status IN ('active','trial','past_due')
  AND  (payment_profile_id IS NULL OR payment_profile_id = 0);
```

## Related

- **Technical Reference**
- **Admin User Guide**