GUIDE

APRIL 4, 2026

ZEROTOUCH DUNNING PLAYBOOK: STRIPE SMART RETRIES + MAKE/ZAPIER (SOLO OPERATOR EDITION)

A practical, copy-ready playbook for solo service operators to configure Stripe Smart Retries, build a webhook-driven outreach flow in Make/Zapier, track recoveries in Notion, and test it all with Test Clocks—so you stop chasing invoices and protect margin automatically.

FROM EPISODE

STOP CHASING INVOICES: ZERO-TOUCH DUNNING WITH STRIPE SMART RETRIES

CONTENTS

What you're building (and with which tools)

Step 1 — Configure Stripe for recovery (the settings that actually matter)

Step 2 — Model the events and set reliable triggers

Step 3 — Orchestrate customer outreach that actually works

Step 4 — Track recoveries in Notion (and make the proof visible)

Step 5 — Handle ACH and non-retryable edge cases

Step 6 — Test end-to-end with Stripe Test Clocks

Step 7 — Harden the system and measure what matters

Build a zero-touch payment recovery system that runs without you: configure Stripe Smart Retries correctly, orchestrate outreach with Make/Zapier, track recoveries in Notion, and validate with Test Clocks. This is the exact operator play Jordan runs to protect margin with no employees.

WHAT YOU'RE BUILDING (AND WITH WHICH TOOLS)

You'll ship a working recovery loop in ~90 minutes using:

- Stripe Billing (subscriptions + invoices)
- Make.com or Zapier (webhook orchestration)
- Slack (internal alerts)
- Email/SMS sender (Gmail/Twilio or equivalents)
- Notion (Payment Recovery tracker + client-visible proofs)

Outcome: >80% of recoverable failed payments are handled automatically within ~5 days, with zero manual chasing and clear visibility on what's still at risk.

STEP 1 — CONFIGURE STRIPE FOR RECOVERY (THE SETTINGS THAT ACTUALLY MATTER)

1. Subscription retries

- Stripe Dashboard → Billing → Revenue recovery → Retries
- Policy: Smart Retries ON, 8 attempts within 14 days
- End-state when all retries fail: set to "Mark as unpaid" (recommended for SMB retainers so access pauses cleanly while remaining reversible)

2. One-off invoice retries

- Stripe Dashboard → Settings → Billing → Invoices → Advanced invoicing features
- Enable retries to mirror your subscription policy (8 attempts/14 days where available)

3. Customer self-service

- Enable Customer Portal and include the payment-method update link in all outreach
- Add card updater and bank account options if you support ACH

Notes

- “Past_due” leaves service running; avoid it unless you intentionally allow grace access.
- For true enterprise exceptions, document a manual review before cancellation/unpaid.
- If you use Stripe Automations, expect `next_payment_attempt` updates to arrive on `invoice.updated` events.

STEP 2 — MODEL THE EVENTS AND SET RELIABLE TRIGGERS

Trigger events to watch (Make: Stripe → Watch Events; Zapier: Stripe → New Event)

- `invoice.payment_failed` (primary trigger; includes `attempt_count`)
- `invoice.payment_succeeded` (or `invoice.paid`) for recovery confirmation
- `invoice.updated` to read `next_payment_attempt` when Automations are enabled
- `payment_intent.payment_failed` for one-off payments

- `charge.failed` as a safety net (rarely fires alone but useful for redundancy)

Fields you'll use

- `invoice.id` — unique key for tracking
- `attempt_count` — drives tone/urgency
- `amount_due`, `customer_email`, `customer_name`
- `next_payment_attempt` (epoch) — schedule awareness
- `payment_settings.payment_method_types[]` — branch card vs ACH

Idempotency & dedupe (don't send three emails for one fail)

- On every event, first check a Data Store (Make) or Storage (Zapier) keyed by `event.id`
- If seen, stop; if new, process and store `event.id`

Security

- If you ever swap to custom HTTP webhooks, verify the `Stripe-Signature` header and pass the raw request body to your verifier. Rotate signing secrets periodically.

STEP 3 — ORCHESTRATE CUSTOMER OUTREACH THAT ACTUALLY WORKS

Router logic (branch on `attempt_count` for cards)

- Attempts 1–2: friendly heads-up via email

- Subject: “Quick fix: your payment didn’t go through (we’ll retry automatically)”
- Body highlights: what failed, when we retry next, 1-click update link `[PORTAL_LINK]`
- Attempts 3–5: clear urgency; add SMS (opt-in only)
  - Subject: “Action needed to avoid service interruption”
  - SMS: “Heads up — we’ve retried your [INVOICE_AMOUNT]. Update card here: [PORTAL_LINK]”
- Attempts 6–8: final notices; include Slack alert to you
  - Subject: “Final attempt today to keep your access active”
  - Internal Slack: “Client [NAME] at attempt [N]/8 on [INVOICE_ID], next retry [DATE].”

Timing tips

- Send customer comms immediately on failure; reference `next_payment_attempt` so expectations are set (“We’ll retry on Tuesday at 9:14am”).
- Respect quiet hours for SMS. Email all attempts; SMS only from attempt 3+.

Links & access

- Always include: `[PORTAL_LINK]`, invoice link, and support reply-to.
- If end-state = unpaid, remind that access pauses after final attempt until payment method is updated.

STEP 4 — TRACK RECOVERIES IN NOTION (AND MAKE THE PROOF VISIBLE)

Create a Notion database: Payment Recovery

- Properties (suggested)
  - Client (Title)
  - Invoice ID (Text, unique)
  - Amount (Number, currency)
  - Attempt Count (Number)
  - Next Retry (Date)
  - Status (Select: In Retry, Recovered, Unpaid, Cancelled, Hard Decline)
  - Failure Date (Date)
  - Recovery Date (Date)
  - Method (Select: Card, ACH)
  - Decline Code (Text)

Automation writes

- On `invoice.payment_failed`: upsert by Invoice ID; set Attempt Count, Next Retry, Method, Decline Code, Status = In Retry
- On `invoice.payment_succeeded`/`invoice.paid`: set Status = Recovered; set Recovery Date
- On final failure with end-state = unpaid: set Status = Unpaid

Proof fields (optional but powerful)

- Payment Reliability Score (%)
  - Formula: `toNumber(prop("Successful Payments")) / toNumber(prop("Total Payment Attempts"))`

- Format as percent; show this in client portal for gentle nudge

- Days to Recover
  - Formula: `dateBetween(prop("Recovery Date"), prop("Failure Date"), "days")`

Views

- “At Risk (Next 72h)” filtered by Status = In Retry and Next Retry within 3 days
- “Recovered (Last 30d)” with average Days to Recover rollup

STEP 5 — HANDLE ACH AND NONRETRYABLE EDGE CASES

ACH behaves differently than cards and should feel different to customers.

Branching rules

- Detect ACH via `payment_settings.payment_method_types[]` or `payment_method_details`
- Retries: up to 2 within 40 days of the original attempt (Stripe-level behavior)

Comms

- First failure (ACH): calm heads-up; emphasize clearing times
    - “ACH can take several days to clear. We’ll retry automatically in five business days. No action needed unless your bank details changed.”
- Second failure (ACH): request update; offer card fallback
    - Provide `[BANK_UPDATE_LINK]` and `[PORTAL_LINK]`

Hard-decline codes (any method)

- For non-retryable declines (e.g., stolen card, revocation of authorization), skip Smart Retry messaging and go straight to a “new payment method required” email with portal link.

STEP 6 — TEST ENDTOEND WITH STRIPE TEST CLOCKS

Use Test Clocks to simulate renewals and watch the full loop work in minutes.

Set up

- Stripe Dashboard → Developers → Test Clocks → New Test Clock (set to today)
- Create a test customer + subscription that renews “tomorrow” on that clock
- Add a failing test card: 4000 0000 0000 0002

Validate

- Advance the clock by 1 day → confirm `invoice.payment_failed` fired
- Check Make/Zap: scenario ran, email 1 sent, Notion record created with Attempt Count = 1
- Advance multiple days → watch Attempt Count grow, messages escalate, Next Retry update

- Flip to a successful test card and advance time → verify `invoice.payment_succeeded` closes the loop and Status = Recovered

Caveat

- Test Clocks use mechanical retry timings; production Smart Retries uses ML for optimal timing. That's expected.

STEP 7 — HARDEN THE SYSTEM AND MEASURE WHAT MATTERS

Operational guardrails

- Access gating: with end-state = unpaid, pause delivery and/or revoke service access until payment method is updated
- Slack hygiene: alert on first failure and final attempt only to avoid noise; add a daily digest of "In Retry" items
- Rate limiting: add a simple "one email per attempt per contact" check to prevent duplicates from replays

Security

- If using custom webhooks, verify signatures and store `event.id` for replay safety; rotate signing secrets quarterly

KPI targets (solo operator baseline)

- Recovery Rate = recovered invoices / failed invoices (target: ≥80–90%)
- Days to Recover (median) (target: ≤5 days)
- Attempt Distribution: most recoveries by attempts 2–4; few beyond 6

- Manual Follow-ups Sent (target: 0)

Runbook for non-recoverables

- If Status = Unpaid for >7 days: send a final closure notice; archive project assets per your MSA; mark CRM as churned and disable access