CHECKLIST · FREE RESOURCE

The Checkout and Payments Review

A structured review for product leaders who own online payments: the metrics to track, decline handling, safe retries, method coverage, 3DS friction, provider failover, reconciliation and fees, with an Excel checklist.

By **Ratomir Jovanovic** · ratomir.com/resources/checkout-and-payments-review

---

This review is for product leaders, founders and heads of payments at companies that take money online, whether through a subscription, a marketplace, an app or a classic checkout. It assumes payments already work. The question it answers is how well they work, where revenue is leaking quietly, and which risks would hurt if a provider, a market or a high-volume day went wrong.

The outcome is a clear picture of your payment funnel from the moment a customer clicks pay to the moment finance can explain the money in the bank account, with owners for each metric and a prioritized list of fixes. Payments sit between product, engineering, finance, support and risk, so they often have no single owner. This review gives the product side enough structure to lead that conversation without needing to be a payments engineer.

How to use this

1. Download the Excel review checklist alongside the PDF guide. Each row has a check, the reason it matters, how to verify it and a priority.
2. Pull at least 90 days of payment data, broken down by payment method, market and provider. Without the breakdown most of the checks cannot be answered.
3. Run the review in one working session with engineering, finance and support in the room. Most findings come from the gaps between those teams.
4. Mark each check as met, partly met or not met, and record the evidence in the notes column: a dashboard link, a query, a test result.
5. Fix P1 items first. They cover double charges, lost payments and invisible failures. P2 items are usually conversion and cost improvements. P3 items matter at scale or in specific markets.
6. Repeat the review twice a year, and before entering a new market or adding a provider.

The metrics to own

A payments operation without agreed metrics argues about anecdotes. Agree on definitions first, because the same word often means different things to different teams. "Success rate" in an engineering dashboard may count only requests that reached the provider, while finance counts settled funds.

Metric	Definition to agree	Cut it by	Why it matters
Authorization rate	Approved authorizations divided by authorization attempts	Method, market, issuer country, provider, new versus returning	The most direct measure of revenue lost at the point of payment
Checkout conversion	Completed payments divided by sessions that reached the payment step	Device, method, market	Captures friction before the authorization is even attempted
False decline estimate	Declines on customers who later pay successfully or have no risk signals	Rule, provider, market	Shows good customers being turned away by your own or issuer risk rules
Cost per successful payment	All payment fees divided by successful payments	Method, provider, market	Reveals whether growth in a channel is profitable
Refund rate	Refunded value or count divided by captured payments	Product, market, reason	Signals product, fulfilment or expectation problems
Chargeback rate	Disputes divided by transactions in the relevant period	Method, market, reason code	Card networks monitor this and high rates lead to fees and programs
Time to reconcile	Days from payment to matched entry in the ledger	Provider, method	Tells you how fast finance can trust the numbers

Measure attempts as well as unique payment intents. A customer who tries three times and succeeds once looks like a 33 percent authorization rate per attempt and 100 percent per intent. Both numbers are useful, but mixing them hides problems.

Declines and how you talk about them

Declines come in two broad kinds. Soft declines (insufficient funds at that moment, issuer unavailable, authentication required) may succeed later or with a different flow. Hard declines (stolen card, closed account, invalid number) will not, and retrying them wastes money and can count against you with card networks. Your system should classify decline codes into these groups and act on the class, not on a generic failure.

The customer message matters as much as the logic. "Payment failed" leaves the customer guessing and drives support contacts. A good message tells them what they can do: check the card details, try another method, or contact their bank. It never exposes fraud signals, and it never blames the customer for something your system did. Review the actual copy shown for your five most frequent decline codes, on every platform, in every language you support.

Retries and idempotency

Double charges are the payment failure customers remember. They usually come from a timeout: the client did not get a response, retried, and the provider processed both requests. The defense is idempotency. Every request that creates or captures a payment carries a unique key tied to the customer's intent, and the provider and your own backend return the original result for a repeated key instead of charging again. Check that keys are generated once per intent, not per request, and that they survive a page reload or an app restart.

Retries themselves need rules. Automatic retries for soft declines on recurring payments can recover meaningful revenue when they are spaced sensibly and aligned with when customers are likely to have funds. Retries on hard declines should stop. Card networks set limits on how often a declined transaction may be retried and may charge for excess attempts, so align your retry schedule with your provider's guidance on current rules.

Payment method coverage

Cards are not the default everywhere. In many markets, bank transfers, real-time payment schemes, local debit schemes or digital wallets account for a large share of online payments, and a checkout that offers only cards will lose customers who never see a reason to try. For each market that matters to your revenue, record which methods customers there expect, which you offer, and the authorization and conversion rate per method. Then decide deliberately. Adding a method has a real cost in integration, support, refunds and reconciliation, so the case should rest on your own market data rather than a general belief that more methods convert better.

3DS and strong customer authentication

In the EEA and the UK, strong customer authentication rules require additional verification for many online card payments, usually through 3D Secure. Other markets use 3DS selectively for fraud reasons. Authentication adds friction and some customers abandon at the challenge. Review where challenges happen, the completion rate of challenges by issuer country and device, and whether you are using the exemptions available to you, such as low-value payments, transaction risk analysis performed by the acquirer, and merchant-initiated transactions for subscriptions. Also check the reverse case: requesting an exemption shifts liability for fraud, so the decision should be explicit and owned.

Provider concentration and failover

A single payment provider is simpler to run and often cheaper on volume, but it means one outage stops all revenue. Ask three questions. What happens to checkout if the provider is down for an hour? How would you know within minutes, rather than from customer complaints? And how long would it take to route traffic elsewhere if you had to? A second provider for failover is a significant project, including token portability, reconciliation and contracts. For some businesses the right answer is a tested

degradation plan, a status message and queued retries rather than full redundancy. Either way, the answer should be written down and tested.

Webhooks and reconciliation

Most payment outcomes arrive asynchronously through webhooks or event notifications. Webhooks can arrive late, more than once, or out of order, and occasionally not at all. Your handlers should be idempotent, should verify signatures, and should not assume order. A scheduled job should compare your records with the provider's records and flag payments that are captured with the provider but not fulfilled in your system, and the reverse.

Reconciliation is where finance meets product. Every payment, refund, chargeback and fee in the provider's settlement reports should match a record in your ledger, with differences explained. If finance reconciles by spreadsheet at month end, mismatches are found weeks after they happened, when they are expensive to investigate.

Fees, support and finance explainability

Payment fees are rarely one number. There are processing fees, scheme fees, cross-border and currency conversion fees, dispute fees, and sometimes fixed monthly charges. Ask whether you can see cost per successful payment by method and market, and whether anyone reviews it. Pricing decisions and method choices made without this view often move volume into the most expensive channel.

Finally, test explainability. Pick three real payments: one successful, one declined, one refunded. Can support explain to the customer what happened, using their own tools, within a few minutes? Can finance trace each one from checkout to the bank statement? If either answer is no, the gaps you find are usually the cheapest fixes in this review.

Worked example

This example is hypothetical. A subscription software company sells in eight countries and runs all payments through one provider. The review shows a healthy overall authorization rate, but the breakdown by market shows one market well below the rest. Digging in, most failures there are authentication challenges abandoned on mobile, and the company is not applying the merchant-initiated exemption to renewals, so customers are challenged every month.

The team also finds that renewal retries run four times on any decline, including hard declines, and that support cannot see decline reasons in their tool. The fixes are prioritized: correct the renewal flagging so exemptions apply (P1), stop retrying hard declines and space soft-decline retries (P1), expose decline categories to support (P2), and assess whether a local bank payment method is worth adding in the weak market (P3). Each fix is given an owner and a metric to watch in the checklist.

Common mistakes

- Reporting a single global authorization rate. Averages hide the market or method that is failing.
- Treating every decline the same way, in retries and in messaging.
- Generating a new idempotency key on each retry, which defeats its purpose.
- Assuming webhooks arrive once and in order.
- Adding payment methods because competitors have them, without per-market data.
- Leaving reconciliation entirely to finance, so product never sees the failures it causes.
- Discovering the cost of a provider outage during the outage.

When not to use this

If you are pre-launch or processing a small number of payments, most of these checks will produce noise rather than insight; pick a reliable provider, use idempotency keys from day one and come back when volume makes the breakdowns meaningful. This review also does not replace a PCI DSS assessment, a legal review of payment regulations in your markets, or a fraud strategy designed with specialists. It is a product and operations review, and it works best as the starting point for those conversations.

Editable template: Review checklist (Excel), available at ratomir.com/resources/checkout-and-payments-review

© 2026 Ratomir Jovanovic. Free to use inside your team. Please link to ratomir.com/resources/checkout-and-payments-review when you share it.