Capacitor + RevenueCat + Stripe entitlement checklist

Use this when a React app wrapped with Capacitor already signs people in with Supabase Auth and sells subscriptions on the web through Stripe — and you want App Store and Play purchases to unlock the same paid access. Tick only what you have verified.

This is a self-check for one stack, not official RevenueCat, Apple, Google, Stripe, or Supabase documentation, and not a substitute for those vendors' current guides. Links below go to the official pages this list was checked against on 22 November 2026. If a vendor changes a behavior, trust their docs over this sheet.

1. App user IDs and aliasing with Supabase Auth

- ☐ `Purchases.configure` runs once with the public SDK key. You do not hard-code an `appUserID` into configure for every install.
- ☐ After Supabase Auth has a session, you call `Purchases.logIn({ appUserID: user.id })` with the stable Supabase user UUID — not email, not a display name.
- ☐ When the Supabase session ends, you call `Purchases.logOut()` so the next person on the device does not inherit the previous entitlement.
- ☐ Switching from one signed-in custom ID to another uses `logIn` with the new id (RevenueCat logs the previous custom id out; no merge). You verified this with two test accounts.
- ☐ In the RevenueCat dashboard, the customer for your test user shows the same custom App User ID as Supabase — not only a `$RCAnonymousID:` placeholder after login.
- ☐ If a purchase happened while anonymous, then `logIn` ran, webhook payloads may show `original_app_user_id` as the anonymous id and `app_user_id` as the custom id, with both in `aliases`. Your backend can resolve either to the same person.

  Official reference: Identifying Customers — https://www.revenuecat.com/docs/customers/identifying-customers

2. Stripe → RevenueCat syncing

- ☐ A Stripe web subscription is attached to the same RevenueCat App User ID the Capacitor app uses after Supabase login. A website payment alone does not create an App Store transaction.
- ☐ External Stripe purchases are imported into RevenueCat (`POST /v1/receipts` with `X-Platform: stripe`, or automatic Stripe server notifications with "track new purchases" enabled as documented).
- ☐ When posting manually, `fetch_token` is the Stripe subscription id (`sub_…`) or Checkout Session id, and `app_user_id` is the Supabase-mapped custom id.
- ☐ Stripe webhook events for RevenueCat are limited to the documented set (subscription create/update/delete, checkout completed, charge refunded, invoice updated). Extra events cause Stripe to disable the endpoint.
- ☐ You know whether RevenueCat registers the Stripe subscription when the invoice is created or when it is paid, and your "web subscriber opens the app" test matches that setting.
- ☐ After a web subscribe, `CustomerInfo` in the Capacitor app shows the expected active entitlement for that signed-in user without a separate "also check Stripe from the phone" path.

  Official references: Track External Purchases — https://www.revenuecat.com/docs/web/integrations/stripe/track-external-purchases · Stripe Server Notifications — https://www.revenuecat.com/docs/platform-resources/server-notifications/stripe-server-notifications

3. Entitlement naming and client gates

- [ ] One (or a few) entitlement identifiers in the RevenueCat dashboard — for example `premium` — map to the App Store, Play and Stripe products that should unlock the same access.
- [ ] The app and any server gate check `customerInfo.entitlements.active['your_id']` (or the REST subscriber entitlements), not product ids and not “any past purchase”.
- [ ] The entitlement string in code matches the dashboard identifier exactly. A typo silently locks everyone out.
- [ ] The client does not award access from a local premium flag. Cached `CustomerInfo` is for UI; the server decision is keyed to the signed-in Supabase user.
- [ ] A `CustomerInfo` update listener (or equivalent) refreshes the UI after purchase, restore, renewal or expiration without requiring an app restart.

  Official reference: Getting Subscription Status — https://www.revenuecat.com/docs/customers/customer-info

4. Restore purchases

- [ ] There is a user-visible Restore control. `restorePurchases` is only called from that action (it can trigger store sign-in prompts).
- [ ] Restore is not a substitute for `logIn`. On a new device you identify the Supabase user first, then restore if the store account should pull receipts onto that customer.
- [ ] You know your project’s restore / transfer behavior setting, and you tested what happens when a receipt is already attached to another App User ID.
- [ ] Programmatic sync (for migrations) uses `syncPurchases` where appropriate, not a background `restorePurchases`.

  Official reference: Restoring Purchases — https://www.revenuecat.com/docs/getting-started/restoring-purchases

5. Webhooks into Supabase (or your backend)

- [ ] A RevenueCat webhook integration posts to your HTTPS endpoint with an authorization header you verify on every request.
- [ ] You return 200 quickly and process asynchronously. Non-200 responses are retried with backoff.
- [ ] Events are deduplicated by `event.id` so retries do not double-grant or double-revoke.
- [ ] For access changes you re-read the subscriber (`GET /subscribers/{app_user_id}`) and store the entitlements RevenueCat reports now — you do not revoke solely because a `CANCELLATION` arrived.
- [ ] `CANCELLATION` vs `EXPIRATION` vs `BILLING_ISSUE` are handled correctly: cancel often keeps access until expiry; billing issues are not an immediate revoke; expiration is applied through the re-read.
- [ ] `TRANSFER` events re-read every id on both `transferred_from` and `transferred_to`. You do not merge those users in Supabase by accident.
- [ ] The path that maps `app_user_id` / aliases to a Supabase user is idempotent and safe if the worker crashes mid-write.

  Official reference: Webhooks — https://www.revenuecat.com/docs/integrations/webhooks

6. Sandbox test cases

- [ ] Purchase in iPhone or iPad sandbox (or Play license testers), then confirm the same entitlement on the web for that Supabase user.
- [ ] Subscribe on the web through Stripe, then confirm the Capacitor app unlocks for that signed-in user without a store purchase.
- [ ] Renewal and cancellation from the store sandbox and from the Stripe side both update access as expected.
- [ ] Refund path: access revokes when RevenueCat says the entitlement is gone (via webhook re-read), not only when a dashboard looks “cancelled.”
- [ ] Restore on a fresh install of the same store account + same Supabase user.

- [ ] Sign out and sign in as a different Supabase user: premium does not leak.
- [ ] You ran at least one of these on a real device, not only in a mocked fixture.

7. Offline and cache states

- [ ] You call `getCustomerInfo()` when the user hits premium content; the SDK may return cached info synchronously and refresh in the background.
- [ ] You understand RevenueCat's offline grace for previously active entitlements (documented as up to three days while offline) and that offerings may not persist across launches the same way.
- [ ] `logOut` clears the SDK cache; you do not rely on stale premium UI after sign-out.
- [ ] Platform server notifications (App Store / Play / Stripe as applicable) are connected so `CustomerInfo` and webhooks update faster than "next app open."
- [ ] You do not treat "empty entitlements while offline" as proof the user never subscribed — and you do not invent a second source of truth on the device to paper over it.

  Official references: Caching — https://www.revenuecat.com/docs/test-and-launch/debugging/caching · CustomerInfo — https://www.revenuecat.com/docs/customers/customer-info

If several boxes stay empty

That usually means identity, Stripe attachment, or webhook reconciliation — not a missing paywall button. A short walkthrough of the most common failure is at https://appcontinuity.com/articles/why-a-stripe-subscriber-hits-a-paywall/

The HTML checklist, with the same items, stays ungated at https://appcontinuity.com/checklist/. If you want this work done for you on this exact stack, the next step is a free fit check: https://appcontinuity.com/#contact

---

AppContinuity is a solo-developer service. RevenueCat, Stripe, Supabase, Apple, App Store, Google Play and Android are trademarks of their respective owners. AppContinuity is independent and is not affiliated with them. This sheet does not collect a score and is not a promise of store approval.
contact@appcontinuity.com · https://appcontinuity.com