Single Bills

Issue a one-off Stripe-collected bill (or record an offline-paid one) for any work that does not flow through a Service Agreement.

Up to date as of v1.32.0

CONTENTS

- What Single Bills is
- When to use Single Bills vs. other modules
- Stripe Connect is required
- Creating a bill
- The bill reference (BILL-YYYY-NNNN)
- What the client receives
- The public payment page
- What happens when the client clicks Pay
- Self-healing reconciliation (Sync from Stripe)
- Marking a bill paid offline
- Resending a bill
- Cancelling a pending bill
- Expiration
- Optional admin fee surcharge
- Accountant CC and Bill-paid CC list
- The Bills list page
- The four status states
- Who can create and manage bills
- Subscription tier and pricing
- Interactions with other modules
- Troubleshooting

What Single Bills is

Single Bills is the way you charge a client for something that is not already wired into another module. A consultation deposit, a separate document review, a small ad-hoc service, an out-of-band fee top-up — any time you need to send 'pay me X dollars' to a named person with a payable link, this is the right place. You type the client's name, email, a short description, and the amount; the platform mints a unique reference, sends the email,

and opens a public Stripe Checkout page on a per-bill token URL. When the client pays, the bill flips to Paid and both you and the client get confirmation emails.

The module noun is 'Bill', not 'Invoice'. The distinction matters in immigration practice: an invoice records services that have actually been rendered, while a bill simply asks for payment. A consultation deposit, for example, is a bill — the work has not happened yet. Tenant-facing surfaces (sidebar entry, page titles, email subject lines) all read 'Bill'. The underlying database table is still named `invoices` for historical compatibility, but you will never see that word in the dashboard, in client emails, or on the public payment page.

When to use Single Bills vs. other modules

Single Bills is the small-surface fallback for one-off charges. Three other modules already collect payment in their own canonical ways, and you should default to those when the work fits.

- If the work is a signed Service Agreement, the platform auto-creates a bill the moment the agreement reaches fully-signed (the 'Bill on Signing' feature in Service Agreements). That auto-bill is itself a row in the Single Bills system and shows up in your Bills list alongside hand-created ones, but you do not create it manually — the SA module does.
- If the work is a Written Consultation ticket, payment is collected by the booking page at the moment of submission (Stripe Connect, same flow as a live booking). Do not issue a Single Bill on top — that would double-bill the client.
- If the work is a paid Booking, the booking POST collects payment via Stripe Connect on confirmation. Same rule — do not issue a separate Single Bill for the same appointment.
- If the work fits none of the above (a one-off document review at a custom price, a top-up for additional disbursements, a quick out-of-band fee), Single Bills is the right tool.

Stripe Connect is required

Single Bills cannot send anything until you have connected a Stripe account through the platform's Stripe Connect flow (Settings ’ Payments). Without a connected account, the New Bill button is disabled and a clear notice on the bills list page points you at the Settings page to finish onboarding. This is not a soft warning — the create route hard-rejects with a 409 if you somehow get past the UI gate, because clients pay through your Stripe account directly, and no account means nowhere for the money to land.

Stripe Connect is free to set up; the platform applies a small commission on top of each transaction (the standard 1% Basic / 0% Premium platform fee) which Stripe takes care of routing on every successful charge. Stripe also takes their own per-transaction fee directly from each payment; the platform never touches the funds in transit. The receipt and the deposit both land in your Stripe account, ready to transfer to your bank on Stripe's standard payout schedule.

Creating a bill

Sidebar ’ Single Bills opens the list page. Click 'New Bill' and a compact form drops down with four required fields and one optional. Client name and client email (required — the bill is addressed to a specific person, and the email is where the pay link goes). Description (optional — short free text shown on the bill page, e.g. 'Document review — passport package'). Amount in dollars (required, must be greater than zero — the platform converts to cents server-side; do not pre-format with the currency symbol). Expiry date (optional — if set, the bill auto-expires after end of day on that date and Stripe cannot collect against it anymore).

Click Send and three things happen in sequence: the platform mints a unique reference (BILL-YYYY-NNNN format, one global counter that resets each January per tenant), inserts the bill row at status 'pending', and sends the bill email to the client through your tenant's branded transport. The response includes the canonical row plus a flag confirming the email went out. The bill immediately appears at the top of the list with a maroon pending pill.

The bill reference (BILL-YYYY-NNNN)

Every bill carries a unique reference of the form BILL-YYYY-NNNN — for example BILL-2026-0042. The reference is global across all bills your tenant has issued, not tenant-slug-prefixed like booking references are. The sequence resets each January, so the first bill of 2027 will be BILL-2027-0001 regardless of where the 2026 counter stopped. The reference is human-readable, appears on every bill email and the public payment page, and is the canonical handle the client should quote if they call to ask about a payment.

> **Note**: Pre-2026-05-14 the prefix was 'INV-' (for 'invoice'). The platform renamed the noun to 'Bill' in Phase 25 alongside the broader dashboard polish; the new prefix matches. Old bills with the INV- prefix continue to render unchanged — the parser accepts both prefixes, and the reference counter walks across both during the cutover so the year-sequence never repeats.

What the client receives

The client gets a single email at the address you typed, sent from your tenant-branded sender ('<Company> via RCIC App'). The subject line carries the bill reference. The body identifies your company, the description (if any), the amount due, the currency, and the expiry date if one was set. The primary action is a maroon 'Pay this bill' button that links to a public per-bill URL at /invoice/<token> on the platform — the token is 32 hex characters of cryptographically secure randomness, so the link is unguessable. Reply-To on the email is your company's reply address (Settings ’ Company) so when the client hits Reply, they reach you and not the platform.

If you have configured an accountant CC (Settings ’ Service Agreements ’ Billing) or a tenant-wide Bill-paid CC list, the accountant address is automatically added to the email envelope so they see every bill that goes out. The accountant CC is one address per firm; the bill-paid CC list is a separate list of up to five addresses that fire only on the paid confirmation, not on the original bill. Both are optional and silent by default.

The public payment page

Clicking 'Pay this bill' opens a public page at /invoice/<token> on the platform domain. The page shows your company name and logo, the bill reference, the description, the amount, the currency, an honest expiry timestamp if one is set, and a single maroon 'Pay now' button. The page is responsive, no login required (the token is the capability), and bilingual when the client's browser locale is detected as fr-CA. There is no other surface on the public page — no list of past bills, no account creation prompt, no way to discover anything else about your tenant. The page is single-purpose: pay this one bill or close the tab.

Once a bill is paid (or cancelled, or expired), the public page continues to resolve at the same URL — it just shows a different state. A paid bill reads 'Payment received' with the paid date. A cancelled bill reads 'This bill has been cancelled'. An expired bill reads 'This bill expired on <date>; contact the firm if you still need to pay'. The token never stops working; the state simply transitions.

What happens when the client clicks Pay

Pay now opens a Stripe-hosted Checkout Session pre-loaded with your bill's amount, currency, and description. Stripe handles the card form, 3-D Secure if required by the issuing bank, error messages, retry on a declined card, and the receipt email. The client never types their card number on the platform; the card data goes directly from their browser to Stripe and never touches the platform's servers. On success, Stripe redirects them back to your bill page with ?paid=1 in the URL, and a Connect webhook fires asynchronously to the platform confirming the payment landed on your connected account.

Two emails fire on a successful payment: the client gets a 'Payment received' confirmation with the receipt-like body restating the bill and amount, and you (the tenant) get a notification email confirming the bill was just paid. Both emails copy the accountant CC if configured. The bill row flips to status 'paid' with the paid_at timestamp stamped, and the Bills list updates with an emerald pill instead of the pending maroon.

Self-healing reconciliation (Sync from Stripe)

Stripe webhooks are reliable but not instantaneous, and on rare occasions a webhook delivery fails — a Vercel cold start that timed out the webhook, a Stripe Dashboard re-routing change, a Connect endpoint mid-rotation. The platform has two recovery paths so a paid bill never stays stuck on pending. First, when the client lands back on the public bill page with ?paid=1 in the URL, the page calls a self-heal endpoint that asks Stripe directly for the most recent Checkout Session and reconciles the row if Stripe confirms the payment. Most webhook misses heal within seconds of the client returning. Second, every pending bill in your dashboard list carries a 'Sync from Stripe' button that runs the same reconciliation manually — useful for the rare case where the client paid but did not return to the bill page (closed the tab, paid through a copied URL, etc.).

> **Note**: The public page never optimistically claims 'Paid' on the ?paid=1 redirect — until the self-heal endpoint confirms Stripe's record matches, the page shows an honest amber 'Payment is processing' card. This is deliberate; an over-eager 'Paid' on a payment that ultimately fails would be more confusing than a brief processing state.

Marking a bill paid offline

Some clients pay outside Stripe — an e-Transfer to your bank account, a cheque mailed to the office, a wire from outside Canada, cash dropped off at the front desk. The bill is still real and still belongs in your records. The Bills list has a per-row 'Mark paid' action that flips a pending bill straight to paid without going through Stripe. Status moves pending ’ paid, paid_at gets stamped, and the same two confirmation emails fire as if the payment had come through Stripe (client gets the 'Payment received' email, you get the tenant notification). The payment_intent_id on the row stays null, which is how Reports + admin dashboards distinguish Stripe-collected revenue from offline-paid bills (the platform earns no commission on offline-paid bills because no money moved through Stripe to charge against).

Mark paid is owner / admin only — staff members do not see the button. Once a bill is marked paid manually, the platform also best-effort expires the open Stripe Checkout session associated with the bill so the client cannot accidentally double-pay by clicking the original email link after you have already recorded their cheque. If the expire fails (Stripe down, network glitch), the manual paid status still sticks; the platform logs the failure for support follow-up.

Resending a bill

If the client says they did not receive the email (caught in spam, deleted, forwarded to the wrong inbox), the Bills list has a 'Resend' action on every pending row. Clicking it re-sends the original email to the same address with the same payment URL — the token does not change, the public payment page is identical, and the previous email link still works (Resend does not invalidate it; both inboxes can route the client to the same Stripe Checkout). Resend is gated server-side on bill status: a paid, cancelled, or expired bill cannot be resent (the action returns an error). The send goes through the same accountant-CC routing as the original.

Cancelling a pending bill

If you realise you sent a bill to the wrong person, with the wrong amount, or for work that ultimately did not proceed, the Bills list has a per-row 'Cancel' action. Cancel only works on pending bills (paid + cancelled + expired bills cannot be re-cancelled; the action returns an error). On cancel, the row flips to status 'cancelled', and the platform best-effort expires the open Stripe Checkout session so a client clicking the pay link mid-cancel cannot still complete the payment. A race protection on the SQL update ensures that if a Stripe payment lands at exactly the same moment as the cancel, whichever wins wins cleanly — the client either pays (Stripe wins) or sees the cancellation (we win), never both.

> **Important**: Cancellation is not the same as a refund. Cancel only stops a pending bill from being paid going forward; it does nothing to a paid bill. If you need to refund a bill that has already been paid through Stripe,

> do it from Stripe directly (Stripe Dashboard ’ Payments ’ the relevant charge ’ Refund). The bill row stays paid on the platform regardless; the refund is a Stripe-side action recorded on Stripe's books.

Expiration

The Expiry field on the create form lets you set a date after which the bill auto-expires. The conversion is end-of-day-in-your-timezone, so an expiry set to '2026-06-30' means the bill stays payable through all of June 30 in your company's local time and flips to expired at midnight on July 1. The status transition is what blocks further payment — the platform tells Stripe to expire the Checkout Session, and the public page renders the friendly 'expired' state.

Expiry is optional. If you leave the field blank, the bill stays payable indefinitely — the platform never auto-expires a bill that does not carry an explicit expires_at value. Set an expiry when the bill is genuinely time-limited (an offer with a deadline, a deposit that loses its hold if not paid by a certain date) or leave it blank when you are happy for the bill to stay open until paid or until you explicitly cancel it.

Optional admin fee surcharge

Some tenants charge a small administration fee on top of the headline amount to cover Stripe's per-transaction cost or as a flat handling charge. Settings ’ Service Agreements ’ Billing carries the controls: enable an admin fee on bills (separate from the booking-side surcharge), choose either a percentage or a flat amount, and the value to apply. Once enabled, every new bill snapshot the admin fee at creation time — the value is recorded on the bill row so the public page can render a clean 'Subtotal + Admin fee = Total' breakdown even if you change the setting later.

The admin fee is yours to keep — the platform does not take any portion of it (the platform commission is calculated on the underlying amount, not on the admin fee). The public page clearly labels the surcharge so the client sees exactly what they are paying for. Existing bills are unaffected by setting changes; the snapshot is locked.

Accountant CC and Bill-paid CC list

Two optional copy lists send a duplicate of every bill email to additional addresses. The Accountant CC (Settings ’ Service Agreements ’ Billing) is one tenant-wide email address that receives a copy of every bill the platform sends — the initial 'please pay' email, the resend, and both confirmation emails on payment. It is the right place to put your accountant's address if they want to see every bill flow in real time. The Bill-paid CC list (same Settings page, separate card) is a list of up to five email addresses that receive a copy of the paid confirmation only — not the initial bill, not the resend. Use this when you have multiple people (a senior accountant, an office manager, a partner) who want to see when money lands but do not need to see every outgoing bill.

Both lists deduplicate against the client address and against each other before sending, so no one receives two copies of the same email. The Accountant CC and the Bill-paid CC list operate independently — you can enable one, both, or neither.

The Bills list page

Sidebar ’ Single Bills opens the list page. Every bill ever created by your tenant (manually or by the SA Bill on Signing auto-creator) shows up here, most recent first, capped at 200 rows for performance. Each row shows the reference, client name, client email, description, amount, status pill, created date, expiry date (if set), and paid date (if paid). Per-row action buttons depend on status: pending bills show Mark Paid, Resend, Sync from Stripe, and Cancel; paid bills show no actions (they are terminal); cancelled and expired bills show no actions either.

The list supports column-header sorting (most recent first by default, click a header to re-sort) and a search box that filters in-memory against the reference, client name, and client email. The search is fast — there is no round trip to the server — so typing 'BILL-2026-004' narrows to the matching row in a single keystroke. For bills older than the 200-row window, ask the operator (or run a Supabase query directly if you have access).

The four status states

- pending — the bill was created and the email was sent; the client has not yet paid. The list renders a maroon pending pill. This is the only state in which Mark Paid / Resend / Sync from Stripe / Cancel actions are available.
- paid — Stripe (or a manual mark-paid) confirmed payment landed. The list renders an emerald paid pill alongside the paid_at timestamp. This is a terminal state; the row cannot transition back.
- cancelled — the operator clicked Cancel on a pending bill. The list renders a charcoal cancelled pill. This is terminal; cancelling is a final decision (re-issue a new bill if you change your mind).
- expired — the expires_at timestamp has passed and the bill auto-expired. The list renders a charcoal expired pill. This is terminal; if the client still wants to pay, issue a new bill with a fresh expiry (or no expiry).

Who can create and manage bills

Single Bills is a per-member module-access permission. By default, Owners and Admins can create, resend, mark paid, sync from Stripe, and cancel bills; staff members can see the list but cannot create or modify rows (the New Bill button does not render, and the per-row action buttons are hidden). The owner of the tenant can change a specific member's access from Settings ’ Team ’ that member's row ’ Module Access checklist — flipping the Single Bills checkbox on grants the staff member full create-and-manage rights for bills.

Module access is enforced both server-side (the API routes 403 if a staff member without the permission tries to call them directly) and client-side (the UI hides buttons the caller cannot use). The permission applies to all bill actions uniformly; there is no finer-grained 'can mark paid but not cancel' split.

Subscription tier and pricing

Single Bills is available on every tier — both Basic and Premium tenants get the full module from day one. The cost model is the same as Bookings: the platform applies a small commission (1% on Basic, 0% on Premium, capped on Basic) on top of each successfully-paid bill collected through Stripe Connect. Offline-paid bills carry no

commission since no money moves through Stripe. There is no recurring fee on the module, no per-bill platform charge, and no cap on the number of bills you can issue.

Stripe charges its own per-transaction fee (typically 2.9% + a fixed amount per charge for Canadian or US cards, slightly higher for international cards) directly out of each payment before the funds land in your Stripe balance. That fee is between you and Stripe; the platform never sees it. If you want the client to effectively cover Stripe's processing cost, that is what the optional admin fee on the bill is designed for.

Interactions with other modules

Single Bills sits underneath several other modules as the shared payment-collection layer. Service Agreements auto-create a bill when an agreement reaches fully-signed if the tenant has Bill on Signing enabled (Settings ’ Service Agreements ’ Billing). The auto-bill is just a Single Bills row with a synthetic description like 'Service Agreement BILL-2026-...' — it shows up in your Bills list, follows every same lifecycle rule (paid / cancelled / expired / resend / sync), and reuses the same client emails. Active File Review's payment request feature works the same way: when an AFR row reaches the post-signed state and the parent SA auto-billed, the AFR module stamps the bill reference back to the AFR row so the matter dashboard can show paid/unpaid at a glance.

Single Bills also feeds the admin Revenue console. Every paid bill (Stripe-collected or marked paid offline) appears in the per-tenant Revenue breakdown, separated into 'Cash in' (Stripe-collected, contributing to the platform's commission) and 'Offline paid' (manually marked, not counting toward commission). The breakdown lives on /admin/revenue and is only visible to platform superadmins, but the underlying signal is what the platform's commission is calculated against. The Store wallet does not interact with Single Bills — Single Bills always collects through Stripe Connect (or records an offline payment), never from your Store credit balance.

Troubleshooting

-

New Bill button disabled / 409 'Connect Stripe before sending invoices' — Stripe Connect is not finished. Open Settings ’ Payments, complete the Stripe onboarding flow, and return. The button re-enables the moment Stripe confirms onboarding is complete.

- Bill stuck on pending after the client paid — first try clicking 'Sync from Stripe' on the row. The platform queries Stripe directly and reconciles. If sync confirms no payment exists on Stripe's side, the client probably did not complete checkout; ask them to try the email link again.
- Client says they did not get the email — first verify the email address on the row is correct (typos happen). If the address is right, click Resend; the platform re-fires the same email through the same transport. Have the client check spam / Junk; tenant-branded mail from new domains occasionally lands there on first delivery. If the email still does not arrive, contact platform support.
- Cannot cancel a bill — only pending bills can be cancelled; the action returns 409 on a paid, cancelled, or expired row. If you need to undo a paid bill, you need to issue a Stripe refund directly from the Stripe Dashboard, not from the platform.
- Race-condition error 'This bill was just marked paid by another process' on Mark Paid — the Stripe webhook arrived at the exact same millisecond you clicked. The bill is paid; refresh the list and the emerald paid pill should already be on the row. The race protection is doing its job; no further action needed.
- Bill marked paid offline but no confirmation emails went out — the platform fires both emails best-effort after a manual Mark Paid, and a transient SMTP failure can drop one or both without rolling back the paid status (the paid record is the source of truth, not the email). If the client needs the confirmation, the simplest path is to forward them a screenshot of the paid bill row from your dashboard. If transient SMTP failures persist, contact platform support.