Written Consultations

Tiered written-advice tickets: client buys a paid bundle, uploads documents and questions, the RCIC replies through a structured back-and-forth with a bounded number of follow-ups and a hard turnaround clock.

Up to date as of v1.29.0

CONTENTS

What Written Consultations is

Written Consultations (WC) is a tiered written-advice product. The client pays up front, uploads supporting documents through a secure portal, types specific questions inside structured fields, the licensee reviews everything and replies with a written answer the client can download as a PDF, and a small bounded number of follow-up round-trips clean up anything that needs clarification before the ticket closes. The whole exchange runs on its own portal — no email back-and-forth, no scheduling, no live meeting. The deliverable is a structured written record both sides can refer back to.

The module is shaped around the operational reality of immigration advice that does not need a meeting. A client with a Procedural Fairness Letter to answer, a specific section of IRPA they want explained against their facts, a refusal letter they want triaged, a PR card renewal eligibility question that hinges on a precise residency calculation — these are all situations where a written, attributable, deliverable-as-a-PDF answer is both more useful and easier to bill for than a 30-minute video call. WC turns those questions into a productised, repeatable workflow with a fixed price, a fixed turnaround, and a bounded scope.

When to use Written Consultations vs. other modules

WC overlaps with three other modules and sits between them on the spectrum of formality + commitment. Bookings is for live conversations (video, phone, in-person) where back-and-forth dialogue is the point. Service Agreements is for full retainers where the licensee takes over a matter, files documents, communicates with IRCC, and bills a structured fee schedule. Active File Review is for one-off responses to a specific government letter that follows a defined three-pathway treatment (Coach Me / Draft for Me / Full Representation). Written Consultations is the right tool when the client wants formal written advice on a defined question or set of questions, but does not need (or want to pay for) a full retainer or a real-time meeting.

- Pick Bookings when the value is the conversation — listening to context, asking follow-ups in real time, reading between the lines.
-

Pick Written Consultations when the question is well-defined and the value is a written, citable, attributable answer the client can reread, forward to a sponsor, or attach to a future application.

- Pick Active File Review when there is a specific government letter or document to respond to (PFL, refusal, RFE) and the workflow needs an intake triage + pathway choice + receipt-tracking flow.
- Pick Service Agreements when you are taking over a matter — the work continues beyond a single answer, the licensee is the named representative, and the billing structure needs an explicit scope + payment schedule.

The 'written' service format

Like every other module on the platform, Written Consultations starts on Services. A WC service is just a regular service row with the format dropdown set to 'Written consultation (asynchronous)' instead of 'Live'. Picking the written format reveals a Tier configuration card with twelve fields that define what the client gets for their payment — how many questions of each kind, how many documents, how many pages of supporting material, how long it should take, whether a rush option exists, and the scope-of-services prose that gets printed verbatim into the agreement the client signs at checkout.

Each WC service is a tier in its own right. Many tenants ship three or four side by side — a Bronze tier with one main question and one document, a Silver tier with two main questions plus a related question, a Gold tier with the higher caps and a context-file upload, an Express tier with a rush turnaround. Every tier is the same database row shape with different configuration values; nothing about the platform privileges a particular tier or a particular naming convention. Switching a service from 'live' to 'written' (or back) clears the off-format configuration fields automatically; the cleanup is enforced at save time so no stale wc_* values linger on a live row.

The twelve tier-configuration fields

Every WC service carries twelve tier fields. They are all set once when you create the service and are snapshotted onto each new ticket at allocation time so a tenant cannot retroactively shrink a tier the client already paid for. The fields decompose into four groups: question caps, document caps, time caps, and the scope-of-services text.

- wc_max_main_questions — the cap on main questions (broad, document-independent). At least one question kind across main / related / doc-specific must be greater than zero.
- wc_max_related_questions — the cap on related questions (follow-on questions attached to a main question; they do not stand alone).
- wc_max_doc_specific_questions — the cap on doc-specific questions (questions attached to a specific uploaded document, e.g. 'what does paragraph 4 of this letter mean for my case'). Nullable; leave blank to disallow this kind entirely.
- wc_max_docs — the maximum number of distinct files the client can upload (must be at least 1).
- wc_max_pages — optional hard cap on total page count summed across every uploaded document. Null = no page cap; non-null = enforced server-side via @cantoo/pdf-lib's PDF page count.
- wc_max_follow_ups — how many follow-up round-trips the client can fire after the licensee delivers the first answer. Bounded; once exhausted, the ticket closes automatically.
- wc_allows_context_file — boolean. When true, the portal lets the client upload a single 'context' file (the matter background, a prior application, a CV) on top of the question-specific document set.
- wc_turnaround_days_min / wc_turnaround_days_max — the business-day window you promise the client. The max becomes the target_response_date stamped on the ticket; the min is honest disclosure on the public agreement.
-

wc_rush_addon_cents — optional rush surcharge in cents. When set, the booking form offers a 'rush' checkbox that adds this amount to the price and overrides the standard turnaround.

- wc_rush_business_days — turnaround when rush is selected (defaults to 1 business day if any rush addon is set).
- wc_scope_text — required, capped at 4000 characters. Printed verbatim into the Initial Consultation Agreement the client signs at checkout; this is where you describe what the tier covers, what it does not cover, and any specific exclusions (e.g. 'this tier does not cover review of full study permit history').

The three question kinds

Questions are not a free-text mush; they are typed by kind, and the portal enforces the per-kind caps at submission time. The three kinds reflect three legitimately different ways an immigration question can sit on top of supporting material.

- Main — a self-contained question that stands on its own and does not rely on a specific document to make sense. Examples: 'am I eligible for an Open Work Permit under the spousal Public Policy', 'does my residency calculation qualify me for citizenship'. Main questions are the backbone of the ticket; the per-kind cap is usually the most generous of the three.
- Related — a question that hangs off a main question and would not stand alone. Example: a main question about Open Work Permit eligibility, with a related question asking what happens if the principal applicant's study permit lapses mid-process. The portal lets clients tag related questions to their parent main question so the licensee sees the dependency at a glance.
- Doc-specific — a question explicitly attached to one of the uploaded documents. Example: an uploaded refusal letter with a doc-specific question asking 'paragraph 4 references section 11(1); is the officer's interpretation correct against my facts'. The portal links the question to the document so the licensee can open both side by

side. Doc-specific questions are optional on a tier; setting wc_max_doc_specific_questions to null disallows them entirely.

Documents and the page cap

The wc_max_docs cap limits the number of distinct files the client can upload. Acceptable file types are the same allowlist the booking-page attachments use: PDF, DOCX, DOC, JPG, PNG. Each file is capped at 10 MB. Files upload through a signed direct-upload URL (Supabase Storage), encrypted at rest under the tenant's data-encryption key alongside every other client-supplied document.

The optional wc_max_pages cap is summed across every uploaded document. When the client uploads a 12-page PDF + a 4-page PDF + 2 JPG photos against a 20-page tier, the server tallies 12 + 4 + 1 + 1 = 18 pages and the upload succeeds. The same set against a 15-page tier would be rejected with a clear error. PDF page counts come from @cantoo/pdf-lib's page-tree walk; non-PDF files (DOCX, images) count as 1 logical page each. Page counting is best-effort — an encrypted or corrupt PDF gracefully degrades to 1 page rather than blocking the upload entirely.

The optional context file

wc_allows_context_file is an opt-in boolean per tier. When on, the portal offers the client a 'Background context (optional)' upload slot above the per-question document slots. The context file is the same shape as a regular document (PDF / DOCX / image, 10 MB cap, 1 page counted by default) but it does NOT count against the per-question doc-specific question cap — it is purely additional background material the licensee can read to set the scene before answering the specific questions.

Tenants typically reserve the context file for higher tiers as a deliberate up-sell — the lower tiers force the client to be precise about what they want answered, the higher tier lets them throw in their full prior application or a long backstory file for the licensee to skim. Whether you enable it is a positioning decision, not a technical one.

Turnaround and the optional rush addon

Two related fields shape the time commitment. wc_turnaround_days_min and wc_turnaround_days_max define a business-day window the licensee promises to respond within (a typical pair is 3 to 5 business days). The max value is what gets stamped on the ticket as target_response_date — the date the client sees on the portal as their delivery deadline. Business days are Monday to Friday in America/Toronto (statutory holidays are NOT excluded in v1; if a Christmas-week ticket has a 5-business-day promise, the count walks straight through any holidays in between).

wc_rush_addon_cents is the optional surcharge that buys a faster turnaround. When set to a positive value, the booking form shows a 'Rush — deliver in N business day(s)' checkbox right alongside the price, and ticking it adds the rush addon onto the line item AND collapses the target_response_date to wc_rush_business_days (default 1). The rush charge is genuinely additional revenue, not a separate tier; the underlying scope is identical, the client is just paying for the licensee to drop everything and turn the answer around faster. If wc_rush_addon_cents is left null, no rush option is offered at all.

Pricing and payment

WC is per-tier flat pricing — the price field on the service is the price the client pays, no matter how many questions they ask within the cap, no matter how many pages they upload, no matter how long the licensee spends answering. There is no per-page surcharge, no hourly billing, no incremental cost for the bounded follow-up round-trips. The whole point of the productisation is that the client knows the total cost the moment they pick a tier; the only variable is whether they tick the rush checkbox.

Payment runs through Stripe Connect at the booking POST — exactly the same flow a paid live booking uses. The client lands on Stripe Checkout, pays with card, and is redirected back to a success page. The platform's standard commission applies (1% on Basic capped at $5000/month, 0% on Premium); Stripe's per-transaction

fee is taken from the licensee's side of the transfer; the rest settles to your connected account on Stripe's regular payout schedule. A 100%-off discount code (see below) skips Stripe entirely.

Discount codes apply to WC

The same booking-side discount codes (Phase 8) that work on live bookings work on Written Consultations. On the booking page, the client can type a discount code; if it validates against your tenant's `discount_codes` table and applies to this service (or all services), the server applies either a percentage or a fixed amount to the line item before passing it to Stripe. The rush addon (if checked) is included in the discounted base — a 50% discount on a $200 tier + $100 rush becomes $150 collected, not $200 collected with the rush taken separately.

A 100%-off discount code (or a fixed-amount code that zeroes the bill out) skips Stripe Checkout entirely and confirms the ticket free-of-charge on submission. The ticket still goes through the full WC lifecycle — the only difference is that the booking row's payment_status is `complimentary` instead of `paid`, and the discount-code usage counter is incremented exactly the same way the live booking path increments it.

The booking flow

On your public booking page, WC tiers render as service cards just like live bookings — name, description, price, duration label (the licensee uses 'duration' to communicate turnaround on WC tiers; many tenants name their WC services 'Standard Written Consultation — 5 business days' / 'Express Written Consultation — 1 business day' so the turnaround is in the title). The client picks the tier, clicks Book Now, and the booking page mounts a written-specific form instead of the live-booking slot picker.

The form is the standard booking form (name, email, phone, optional preferred language) with a few WC-specific additions: the rush checkbox (when offered), the discount code field, and an inline rendering of the wc_scope_text and tier caps so the client sees exactly what they are buying. Submitting the form creates a `booking` row in payment_status='pending' plus a parallel `written_consultations` row in status='pending_payment', then either

redirects the client to Stripe Checkout or (on a 100%-off discount) confirms immediately and fires the welcome email with the portal link.

The welcome email and portal link

On a successful payment (or a 100%-off comp), the client receives a welcome email from your tenant-branded sender ('<Company> via RCIC App'). The email confirms the tier they purchased, the price they paid, the turnaround window, and the target response date. The primary action is a maroon 'Open your secure portal' button that links to a private per-ticket URL on the platform at /portal/wc/<token>. The token is 32 bytes of cryptographically secure random data (43 characters base64url-encoded), and it is the capability — anyone with the URL can open the portal. The portal is bilingual; if your tenant has Premium and has additional client-page languages enabled, the portal renders in any of them via the runtime translation pipeline.

Inside the portal, the client first sees an email-gate: type the email address you used at booking to confirm you are the right person. The check is server-side (POST against the ticket's stored email), case-insensitive, and rate-limited to prevent abuse. Once the email matches, the portal unlocks and the client lands in the appropriate view for the ticket's current lifecycle state — typically the submission view, where they upload documents and type questions.

The lifecycle states

A WC ticket walks through a deterministic state machine. Every state has a single canonical reason it can be in that state, a single set of valid actions, and a single next-state target on each action.

- pending_payment — the booking row was created but Stripe has not yet confirmed payment. Limbo state; if Stripe never confirms (client abandoned checkout), the ticket eventually times out and the cleanup cron removes the orphan booking.
-

awaiting_submission — payment confirmed; the client has the portal link and is expected to upload documents + type questions. This is the entry-into-the-flow state, and it is where the Day-2 / Day-6 / Day-7+ cron reminders fire from.

- submitted — the client clicked Submit. The licensee gets a notification email + a 30-minute review block on their calendar pinned at 8 AM the next business day. The clock on target_response_date is now ticking from this moment.
- under_review — the licensee opened the dashboard detail page and clicked 'Start review'. Cosmetic state mainly; tells the client (and the licensee herself) that work has actually started, and stops the Day-2 reminder from firing again if it had not yet.
- answer_drafted — the licensee finalized the written answer. The client gets the 'your answer is ready' email with the portal link; opening the portal shows the answer in HTML on screen + a Download PDF button.
- clarification_requested — the licensee paused the answer to ask the client for more info before drafting. The client sees a clarification view in the portal with the licensee's question and an inline reply box. The clock can pause here (operator choice — the dashboard surfaces an explicit 'pause turnaround clock' toggle).
- follow_up_submitted — after the answer was delivered, the client clicked 'Ask a follow-up' and submitted a clarification of their own. The state increments follow_up_count_used; once it equals tier_max_follow_ups the next portal page hides the follow-up button.
- closed — the ticket is terminal. Closed by the licensee manually OR by exhausting the follow-up cap OR by the Day-7+ expiry cron if the client never submitted. The portal still resolves at the token URL; the answer PDF remains downloadable indefinitely; no more state transitions are possible.
- cancelled — the licensee cancelled the ticket. Refunds (if applicable) fire through Stripe Connect. Terminal.
-

expired — the Day-7+ cron caught a ticket sitting in awaiting_submission past its target_response_date. No work was ever started; the booking is marked expired and the licensee gets a notification. Terminal.

Submitting questions and documents

In the submission view, the client sees one section per question kind the tier allows. Main questions render as a textarea grid up to wc_max_main_questions; related questions render under each main question (with a per-related-question parent dropdown so the client can tag which main question the related one hangs off); doc-specific questions render under each uploaded document. The 'Background context (optional)' upload slot appears at the top when wc_allows_context_file is on. The per-document upload slots appear one per question that needs a document; clicking each slot opens the file picker with the file type allowlist.

Clicking Submit fires three things in sequence: each document uploads to the portal upload endpoint with a per-file size + MIME + page-count check; the questions JSON gets validated against the tier caps; the ticket transitions from awaiting_submission to submitted, target_response_date is locked in based on submission time + the tier's max turnaround, and three notifications fire — a confirmation email to the client, a notification email to the licensee, and a 30-minute review block on the licensee's calendar at 8 AM the next business day. The portal then switches to the in-progress view, which is what the client sees on every subsequent visit until the answer is ready.

The dashboard list and detail pages

Sidebar ’ Written Consultations opens the list page. Every ticket your tenant owns shows up here, most recent first, capped at 200 rows. Columns are the ticket reference (WC-YYYY-XXXXXX), client name, service name, status pill, target response date, rush flag, follow-up count used out of max, owner member (in Team Mode). The list supports Team Mode owner-scoping — staff see only their own tickets by default, the Owner sees everything with an inline filter dropdown to scope by member.

Click a row to open the detail page. The detail view shows the client identity (linked back to the booking row), the full questions JSON expanded into readable sections (main / related / doc-specific separated), every uploaded document with download buttons, the answer textarea once you start drafting, the lifecycle action buttons (Start review / Request clarification / Save draft / Finalize answer), and a portal-revoke toggle for the rare 'wrong-person-got-the-link' case. The action buttons gate on status — Finalize is only available after a draft exists, Request clarification is only available before the answer is finalized, and so on.

Drafting and finalizing the answer

The answer is composed in a rich-text editor on the detail page. The editor supports standard formatting (headings, bold / italic, lists, links). Save Draft persists what you have written without releasing it to the client — useful for staged work where you draft today and finalize tomorrow. The client sees no change while the answer is in draft; the in-progress portal view continues to show 'your consultant is reviewing your submission' status until you click Finalize.

Finalize Answer is the publish action. The platform parses the HTML answer into a structured react-pdf document, generates a downloadable PDF with your tenant branding + the licensee's name + the ticket reference + the question-and-answer structure preserved, stores the PDF in a private bucket, and emails the client with a 'Your answer is ready' notification. The portal flips to the answer view; the client can read the HTML inline, download the PDF, and (if any follow-ups remain on the tier) click 'Ask a follow-up' to use one of their bounded round-trips.

Asking the client for clarification

Sometimes the submitted questions are not quite clear enough to answer correctly — the client referenced a document they didn't actually upload, the residency dates don't add up, the matter description is ambiguous about which family member's status is in question. The dashboard detail page carries a 'Request clarification' button that pauses the answer-drafting and sends a structured note to the client. The portal flips to the clarification view;

the client sees your message and a reply box, types their reply, and clicks Submit. The licensee gets a notification email; the ticket transitions back to under_review with the new context appended; you draft the answer with the full picture.

Clarification requests fire BEFORE the answer is delivered and do not count toward the follow-up cap. The clarification round-trip is part of getting the first answer right; the follow-up cap is for post-answer continued conversation. Tenants typically use clarification freely (it costs nothing) and follow-ups sparingly (each one is one of the bounded N the client paid for).

Follow-ups after the answer

Once the answer is delivered, the client has up to wc_max_follow_ups bounded round-trips with the licensee. Each follow-up is a new structured back-and-forth: the client types a clarification question in the portal, the licensee gets a notification + a new review block on their calendar, the licensee replies, the answer expands. The platform appends each follow-up to the original answer PDF as a new section so the downloadable record stays comprehensive.

When the cap is exhausted (the wc_max_follow_ups-th follow-up gets answered), the portal hides the Ask-a-follow-up button on the next visit and shows a friendly message: 'You have used all the follow-ups included in your tier. To continue, please book a new consultation.' The ticket flips to status closed automatically. Tenants can also manually close a ticket before the cap is reached (Phase 24F follow-up — explicit Close button on the detail page) and reopen a closed ticket (mig 060) if the client comes back with related questions they realised after the fact.

Manual close and reopen

The detail page carries a 'Close ticket' button visible from any non-terminal state (pending_payment / awaiting_submission / submitted / under_review / answer_drafted / clarification_requested / follow_up_submitted). Clicking it transitions the ticket to status closed and stamps a `pre_close_status` column on the row so the platform

remembers what state the ticket was in before close. This is useful when, for example, the licensee delivered the answer and the client is happy without using any of the follow-ups — closing manually frees up the dashboard list and signals the ticket is done.

Reopening a closed ticket is also supported, with one nuance. The platform reverses the status back to whatever pre_close_status was stamped at close time. If the ticket was closed from answer_drafted, reopen returns it to answer_drafted and the client can use any remaining follow-ups. Reopening does NOT auto-reverse any refund that fired during a cancel — refunds are Stripe-side actions and need to be re-billed offline if the client and licensee agree to continue. The reopen email warns the client of that explicitly: continuing means a new arrangement, not a do-over of the original payment.

Cancellation and refunds

The detail page carries a Cancel button that transitions any non-terminal ticket to status cancelled. The cancel route is sensitive to where the ticket is in the lifecycle. From pending_payment / awaiting_submission / submitted / under_review / answer_drafted / clarification_requested / follow_up_submitted, the cancel offers a full Stripe refund of the booking — the platform queries Stripe Connect for the captured payment_intent_id and fires a refund for the full amount (minus any platform commission already taken). The booking row's payment_status flips to refunded; refunded_amount_cents records the dollar amount returned.

Cancelling a ticket also revokes the portal token (the client cannot open the portal anymore — they see the 'this consultation has been cancelled' blocked view), tears down the calendar review block if one exists, and sends a cancellation email to the client. If the licensee wants to cancel without refunding (the work was done in good faith but the client is requesting a refund the licensee disputes), the platform still flips the status to cancelled and clears the portal but does NOT auto-fire the Stripe refund. The dispute resolution is then between the licensee and the client; Stripe Dashboard remains the source of truth for whether money actually moved.

The Day-2 / Day-6 / Day-7+ cron sweep

A daily cron job (`/api/cron/wc-sweep`) walks every ticket in awaiting_submission and does three idempotent passes. The Day-2 nudge: tickets whose target_response_date is still more than 5 days away AND whose awaiting_submission_reminder_sent_at is null get a portal-link reminder email. The Day-6 warning: tickets whose target_response_date is tomorrow or sooner AND whose expiry_reminder_sent_at is null get an 'expiring soon' email. The Day-7+ expiry sweep: tickets whose target_response_date has already passed are flipped to status expired, the calendar review block is torn down, an expired email goes to both the client and the licensee.

Every reminder email rotates the portal token before sending — the email's link supersedes whatever the client had in their inbox before. This is a deliberate security choice: a stale email from week one is invalidated the moment the Day-2 reminder fires, so a forwarded-by-accident or leaked-by-email-compromise older link cannot still resolve the portal. The reminder cap-of-one-per-stage prevents the cron from spamming a client who is just slow to respond.

Who can create and manage WC tickets

Written Consultations is per-member module access. By default, Owner and RCIC-seat members can see and manage WC tickets; Assistant-seat members cannot (the default is off because the work is regulated immigration advice — only the licensee should be drafting answers under their name). The owner of the tenant can override per-member: Settings ’ Team ’ that member ’ Module Access checklist ’ toggle Written Consultations on to grant a specific Assistant access to the module (rare but possible — for example, an Assistant who is helping with operational follow-ups while the licensee drafts the substantive answer).

Module access is enforced both server-side (the dashboard list page redirects assistants without the permission back to /dashboard; every API route 403s on cross-permission access) and client-side (UI buttons that would not work are hidden). In Team Mode, ownership also matters: a staff member with module access can only mutate

WC tickets where they are the assigned owner_user_id (or the ticket has no owner — shared services). The Owner can mutate every WC ticket regardless of owner_user_id.

Team Mode and per-service ownership

If you are a Premium tenant with multiple team members, each WC service can carry an owner_user_id pointing at the member who delivers that tier. The owner's calendar gets the 30-minute review block on submission. The owner's name (display_name on company_members) prints onto the answer PDF as the named licensee. The owner is the default Reply-To on every client-facing email associated with the ticket. If a tenant offers tiered WC services where different members specialise in different areas (e.g. a junior associate handles standard tier tickets, the senior partner handles Express tier tickets), assigning a specific owner per service routes the work correctly out of the gate.

On a non-Team-Mode tenant, or for a WC service with no assigned owner, the ticket resolves through the company-wide calendar + the Owner member's identity for the answer PDF signature. The same pattern Bookings uses; same three-branch resolver (per-service owner ’ company-level ’ Owner-member fallback).

Subscription tier and pricing

Written Consultations is available on every subscription 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 capped at the monthly free tier, 0% on Premium) on top of each paid ticket collected through Stripe Connect. There is no per-ticket platform fee, no monthly subscription fee on the module itself, no cap on the number of WC services you can publish, and no cap on the number of tickets you can receive. Stripe charges its own per-transaction fee directly out of each payment (typically 2.9% + a fixed amount per charge for Canadian or US cards).

Premium does unlock one related Premium-only capability: runtime client-page translation. If your tenant has Premium and has enabled additional client-page languages (Settings ’ Branding ’ Client page languages), the public booking page renders WC tiers in any of the enabled languages, and the WC portal does the same — a client whose interface language is Mandarin will see your tier description, the questions form, and the answer view in Mandarin, while your draft and the answer PDF stay in English (or French, depending on what you wrote). The translation is on-demand and machine-assisted; the underlying legal text on the agreement the client signs remains in English / Quebec French for legal force.

Interactions with other modules

Written Consultations sits cleanly alongside Bookings and Stripe Connect — the booking POST route handles both formats through one shared payment flow, the dashboard shares the same booking + booking-reference vocabulary, and discount codes flow through both worlds identically. WC results often lead into bigger modules. A client who buys a WC ticket, gets the answer, and decides they want full representation is a natural pipeline into Service Agreements: the WC answer PDF is excellent context for the SA Builder, the licensee has a documented foundation for the new matter, and the SA's Bill on Signing flow handles the retainer billing. If the WC question turned out to be about a specific government letter (a refusal, a PFL), Active File Review is the right downstream module — close the WC ticket as completed, open an AFR matter, and the licensee has both records linked through the same client identity.

WC does NOT auto-create downstream records (no Transfer Room is provisioned, no Service Agreement is seeded, no client portal session shares across modules). Each module is intentionally independent so a WC ticket can stand on its own without committing the licensee to a larger relationship. If the client and licensee decide to keep working together, the upgrade is a deliberate next-module decision, not a default behaviour.

Troubleshooting

- Client says they cannot open the portal — first check that the email gate matches the address they typed at booking. The portal verifies case-insensitively, so 'JANE@EXAMPLE.COM' typed at booking and 'jane@example.com' typed at the portal are the same. If the addresses genuinely differ (the client booked under a typo'd address), the dashboard detail page has an 'Update client email' action that flips the gate target.
- Client says the portal link is dead — every reminder email rotates the token. The most recent email is the only one with a working link; older ones are invalidated. Send a Resend from the detail page; the client gets a fresh email with a working token.
- Submit blocked by 'page cap exceeded' — the sum of pages across uploaded documents is over wc_max_pages. The client either needs to drop a document, replace a long one with a shorter version (e.g. just the relevant pages of a long refusal letter), or upgrade to a higher tier with a more generous page cap.
- Cancel did not refund the client — the cancel route only auto-refunds when the booking is paid through Stripe AND the licensee chose 'cancel with refund' on the cancel modal. If 'cancel without refund' was chosen (or the booking was paid through a 100%-off discount code), no refund fires. Issue a manual refund from the Stripe Dashboard if needed.
- Follow-up button missing in the portal — most often the follow-up cap is exhausted (follow_up_count_used equals tier_max_follow_ups). The portal hides the button when the cap is reached. The dashboard detail page shows the same counter so you can confirm. If the client paid for more follow-ups than the tier allows by mistake (you under-configured the tier), the right resolution is a separate WC ticket on a higher tier — the platform does not retroactively expand a tier mid-flight.
- Ticket expired but client says they tried to submit — the Day-7+ cron only flips the status when target_response_date has actually passed. If the client claims they submitted before expiry, check submitted_at on the dashboard row; a non-null timestamp means the submission actually landed (and the ticket should be in

submitted/under_review, not expired). If submitted_at is null and the ticket expired, the submission attempt did not reach the server (network issue, browser crash, etc.). Reopen the ticket to allow them to retry.