Co-Counselling Agreements

Premium two-RCIC instrument for matters where you co-counsel with another consultant. Shared template library with Service Agreements, AI Populate from uploaded documents, two-party portal signing, encrypted signed PDF, and a one-click bridge that auto-generates a prepopulated client Service Agreement when you are the lead RCIC.

Up to date as of v1.32.0

CONTENTS

- Overview
- Important: read before using
- The CCA list page
- Start a new CCA
- Who is the lead RCIC?
- Invited RCIC: external or RCIC App tenant
- TN-consent profile share (RCIC App tenant path)
- Incoming consent inbox (the responding tenant's view)
- The CCA Builder
- Parties and fee arrangement
- Client and family members
- Matter, scope, and service components
- Role matrix
- The validator
- Templates shared with Service Agreements
- AI Populate
- Preview HTML and PDF
- Send for signature
- The invited RCIC's portal
- Countersign and finalize
- Auto-generated client Service Agreement
- CCA Settings (shared with Service Agreements)
- Reminders and the daily cron
- Encryption, storage, and the audit ledger
- Integrations with other modules

Overview

Co-Counselling Agreements (CCA) is the module you use when you and another RCIC agree to work together on a shared client matter. Examples: you have the language fit but lack regional expertise in the destination province; another consultant has the specialty but you carry the client relationship; you and a colleague split workload on a complex multi-applicant retainer; you bring a junior RCIC into a matter for capacity reasons. In every case the legal instrument that documents the arrangement is the CCA. Unlike a Service Agreement, the CCA is signed between two RCICs — the client does not sign it. The client signs a separate Service Agreement that names both RCICs and references the CCA's terms.

The module sits next to Service Agreements in the dashboard and deliberately shares architecture with it. The 22 CCA clauses are matched per-clause to the relevant SA clauses so the legal vocabulary stays consistent between the two-RCIC instrument and the client-facing instrument. The same template library powers both: when you publish a Service Agreement template, it becomes available in the CCA builder too, and the scope, milestones, payment schedule, and typical professional fee carry across. AI Populate uses the same Claude extractor as SA, so dropping a passport and a refusal letter into the upload widget surfaces the same recommended fields in both modules.

> **Premium**: Co-Counselling Agreements is Premium. Basic tenants see the sidebar entry and the dashboard tile; clicking lands them on a Premium upsell card with a direct subscribe link. Once subscribed, every feature in this chapter unlocks, including the shared SA template library, AI Populate, the two-party portal signing flow, the encrypted PDF storage, and the auto-generated client Service Agreement when you are the lead.

What ships in this module

- A two-decision setup form that captures who is the lead RCIC and whether the invited RCIC is an RCIC App tenant or an external practitioner.
-

A narrow profile-share consent flow that runs when the invited RCIC is an RCIC App tenant, so their identity (name, email, CICC number) reaches your draft only after they have explicitly approved this one CCA.

- An SA-style builder with autosave (800 ms debounce per slice), sticky validator rail, 22 bilingual clauses, parties, family members, service components, fee arrangement, role matrix, and HTML and PDF previews with a DRAFT watermark.
- Template library shared verbatim with Service Agreements, including tenant templates, global templates, and the All / Your templates / Global tab picker with counts.
- AI Populate with Claude Sonnet 4.6, reading uploaded documents (passport, refusal letter, prior agreement) and proposing scope, family members, service components, and a recommended template with chosen-fee MoneyInput.
- Two-party portal signing: the invited RCIC signs via a tokened public link; you (the initiating RCIC) countersign from the dashboard lifecycle view.
- Owner-password protected signed PDF with multi-line audit footer (signer, redacted IP, token-hash prefix, UTC stamp) on every page, encrypted under your tenant DEK before storage.
- Lead-side bridge: when you are the lead RCIC and the CCA reaches completed, a prepopulated client Service Agreement is created automatically with the invited RCIC pre-loaded as co-counsel.

Important: read before using

> **Important**: The CCA is signed between two RCICs only. The client does not sign it and does not become a party to it. The legal instrument that binds the client to the joint engagement is the separate client Service Agreement that names both RCICs as joint counsel and references the CCA's fee-share and responsibility terms. The platform produces the client SA automatically when you (the initiating tenant) are the lead RCIC;

> when the invited RCIC is the lead, that RCIC produces the client SA from their own dashboard, using their tenant's SA Builder.

Client informed consent is a precondition. CICC's Code of Professional Conduct requires that the client knows about and consents to the joint arrangement before either consultant takes any action. The New CCA form makes this explicit with a required affirmation checkbox: you confirm that you have informed your client about the proposed co-counselling arrangement and obtained their consent. The platform does not collect that consent from the client — it is your responsibility, recorded in your file notes, before the CCA leaves your dashboard.

Like every legal-instrument module on the platform, CCA is a workflow tool. It does not give legal advice, it does not substitute for licensee judgement, and it does not create a co-counselling arrangement on its own — only the executed CCA does that, and only your professional review of every clause before signing satisfies your CICC obligations. The 22 clauses are CICC-aligned drafts curated by the platform; you bear final responsibility for the document you sign.

The CCA list page

The list page at /dashboard/co-counselling shows every CCA your tenant has created, ordered most recently updated first. The header carries the module name, a one-line description, and a New CCA button visible to Owner and Admin roles. The table has four columns: Reference (CCA-YYYY-XXXXXX, a maroon link to the detail page), Status (a localized label drawn from the lifecycle enum), Lead (We are the lead RCIC / The invited RCIC is the lead / Unassigned), and Last updated (formatted in your tenant timezone).

Status values in order

- Draft: you are building the CCA. Edits autosave; the builder is in full edit mode.
- Waiting for profile-share consent: the invited RCIC is an RCIC App tenant and you have requested their narrow profile share, but they have not yet accepted or declined.

- Ready to send: the validator is clean and the draft is ready to send to the invited RCIC for signature.
- Sent for review: the invitation email is out; the invited RCIC has a portal link and a pending decision (sign, decline, or request changes).
- Changes requested: the invited RCIC submitted a change request. The lifecycle view shows the request notes; you re-open the draft, edit, and resend.
- Declined: the invited RCIC declined. You can re-open the draft, change terms, and resend to the same or a different invited RCIC, or cancel.
- Completed: both RCICs have signed. The signed PDF is rendered, encrypted, and stored. Both RCICs receive the completion email with the PDF attached.
- Service Agreement draft created: you were the lead RCIC; the platform has auto-generated a prepopulated client Service Agreement draft. The lifecycle view links directly to it.
- Cancelled: you cancelled the matter at any prior stage with a recorded reason.

Start a new CCA

Click New CCA on the list page to open the setup form. The form is intentionally short and asks two yes/no decisions plus a minimal client identity, because the rest of the document is best edited in the full builder once the row exists. The two decisions: Who is the lead RCIC on this matter? (radio: We are the lead / The invited RCIC is the lead) and Is the invited RCIC an RCIC App tenant or an external practitioner? (radio: RCIC App tenant / External). The client identity block asks for the client's full name and email so the row has a meaningful display before the builder starts. The last field is a required affirmation checkbox: 'I have informed my client about this co-counselling arrangement and obtained their consent.' Submit is disabled until every field is valid.

On submit, the platform creates the cca_agreements row with a fresh CCA-YYYY-XXXXXX reference, encrypts the client name and email under your tenant DEK, records the client-consent affirmation timestamp in the audit ledger, and navigates you to the builder. If the invited RCIC is an RCIC App tenant, the status starts at Draft and the builder shows a prompt to request the profile share (see below). If the invited RCIC is external, the status starts at Draft and the builder asks you to fill in the external RCIC's name, email, and CICC number manually.

Who is the lead RCIC?

The lead RCIC is the one who owns the client relationship: the consultant the client first hired, the one who issues invoices, the one whose name appears as primary on the IRCC Use of Representative form. The lead is not necessarily the consultant doing the most work — fee share and workload split are negotiated separately and recorded in the fee arrangement section of the CCA. The lead decision matters for one downstream behaviour: when the lead RCIC's tenant signs the CCA, the platform auto-generates a prepopulated client Service Agreement draft for that tenant. The non-lead RCIC does not get an auto-generated SA (their tenant is not the one with the client relationship).

When you pick 'We are the lead' on the setup form, the platform will run the SA bridge at completion on your side. When you pick 'The invited RCIC is the lead' and the invited RCIC is an RCIC App tenant, the SA bridge runs on the invited tenant's side instead — they see the auto-generated SA on their dashboard, not yours. When the invited RCIC is external, no SA bridge runs (we cannot auto-write to a tenant we do not host); the external RCIC produces the client SA outside the platform per their own workflow.

Invited RCIC: external or RCIC App tenant

The second decision on the setup form is whether your co-counsel is an RCIC App tenant or an external practitioner. When the invited RCIC is an RCIC App tenant, the platform can verify their CICC identity from their tenant profile and route the signing flow through their dashboard — they see the inbound CCA in a dedicated

incoming surface, accept your profile-share request through one click, sign through their authenticated session (no external portal token needed), and receive completion emails branded by your firm. When the invited RCIC is external, the platform treats them as an anonymous third party: you type their name, email, and CICC number manually in the builder; they receive an invitation email with a tokened public portal link; they sign through the portal without any tenant session.

Both paths produce the same legal instrument and the same signed PDF. The differences are operational: the RCIC App tenant path is friendlier for both sides (no manual CICC-number typing, no token to track, the SA bridge can run on either side, the audit ledger is richer because both tenants' identities are platform-verified), but it only works when both consultants happen to be on the platform. The external path is the universal fallback that works with any RCIC anywhere, at the cost of one extra invitation email and a manual identity-entry step on your side.

TN-consent profile share (RCIC App tenant path)

When the invited RCIC is an RCIC App tenant, the builder asks for their tenant code. The tenant code is the short numeric identifier we assign every tenant on signup; your invited RCIC can find theirs on their /dashboard/profile page. The builder parser accepts the code as bare digits, with a TN prefix (TN1234), or with a TEMP prefix (TEMP1234) — the latter is for tenants whose code has not yet been finalized by our team. On submit, the platform sends a narrow profile-share consent email to the invited tenant's Owner and the matter status flips to Waiting for profile-share consent.

The profile share is narrow on purpose. Accepting does not grant blanket access to the invited tenant's data, does not connect the two firms organizationally, and does not share contacts, calendars, files, or any other client data. It shares exactly four fields scoped to this one CCA: the invited RCIC's display name, their professional email, their CICC number, and (when set) their typed-signature font preference for the signing flow. The share burns down with the matter — when the CCA is cancelled, declined, or reaches a terminal status, the profile-share record is closed and no future CCA can reuse it. A new CCA between the same two tenants requires a fresh consent.

Incoming consent inbox (the responding tenant's view)

When another tenant requests your profile share for a CCA, you find the request at /dashboard/co-counselling/incoming-consent. The page shows one card per pending request with the initiating tenant's name, the requested-at timestamp, the CCA reference, and two buttons: Grant (the platform attaches your verified identity to that CCA and emails the initiating RCIC that they can continue building) and Decline (the matter on the initiating side flips to a declined-profile-share state; they can either resend a fresh request after fixing whatever made you decline, or proceed with the external-RCIC path using your name and CICC number typed manually). The page is owner-and-admin only.

> **Note**: Decline a profile-share request when you do not recognize the initiating tenant, when the matter is not one you have agreed to co-counsel, or when you have any concern about the arrangement. Grant only when you have already had the conversation with the initiating RCIC out of band and you understand the matter. The Grant audit-ledger entry on your side and theirs records the decision permanently.

The CCA Builder

The builder at /dashboard/co-counselling/<id> mirrors the Service Agreement builder by design. The layout is a two-column grid: a tall left column holding the editable sections in a vertical stack (parties, client, family members, matter, scope, service components, fee arrangement, role matrix, plus the 22 clause sub-sections), and a sticky right rail holding the validator, the Preview PDF and Preview HTML buttons, the AI Populate button, and a jump-to-section search box. Edits autosave per slice with an 800 ms debounce; a small status pill above the rail reads All changes saved, Unsaved changes, Saving..., or Save failed depending on the autosave queue.

The 22 CCA clauses each have a default body curated by the platform and editable per-CCA. Locked clauses (the parties block, the signature block, the entire agreement, and the four primary structural anchors) cannot be removed; every other clause has a remove button if it does not apply to your matter. Each clause's body

is rich-text-editable inline, with a Reset to default button if you want to discard your edits. The 22 clauses cover: parties (initiating and invited RCIC identification), client identification, matter and scope, fee arrangement, fee share, payment routing, file handling, communication and joint files, joint responsibility allocation, conflicts of interest, confidentiality, electronic communication, dispute resolution between counsel, termination of the co-counselling arrangement, severability, governing law, entire agreement, and the signature block.

Parties and fee arrangement

The parties section holds two cards: Initiating RCIC (your firm — display name, CICC number, professional email, prefilled from your tenant profile) and Invited RCIC (the second consultant — populated from the profile-share for the tenant path, or typed manually for the external path). Both cards expose the same set of editable fields; you can override the prefilled values per-CCA without changing your tenant profile. The role on each card defaults to RCIC; the legal text uses the role label verbatim, so changing it (for example to RISIA) changes the rendered clause text accordingly.

The fee arrangement section documents how the two RCICs split fees. It carries four fields: total professional fee (the amount the client pays), initiating RCIC share (dollar amount), invited RCIC share (dollar amount), and a free-text notes field that explains the rationale (work-share, capacity, complexity, retainer-vs-success structure, etc.). The two share amounts must sum to the total fee; the validator emits a blocker if they do not. The Auto-compose notes button (visible when the notes field is empty) drafts a sentence based on the dollar split for you to edit; it is idempotent — re-running it does not overwrite manual edits, only fills an empty field.

Client and family members

The client section holds the same client identity fields the SA builder uses: full name, email, phone, address, date of birth, country of birth, citizenship. The fields prefill from the new-CCA setup form (just name and email there);

you complete the rest before sending for signature. Every field is encrypted under your tenant DEK at save time, with a blind-index hash on the email for exact-match lookups.

Family members covers spouses, dependent children, sponsors, and any other applicants who appear in the same matter. Each row carries full name, relationship to the main client, date of birth, country of birth, citizenship, and a checkbox for 'shares the matter scope' (used to drive the joint-applicant rendering in the matter clause). Family-member add automatically sweeps the service-components section: any component that names a family member by full name (parentheticals stripped — so 'Maria Lopez (spouse)' matches 'Maria Lopez') gets its familyMemberId resolved retroactively, so the validator stops complaining about unlinked components.

Matter, scope, and service components

The matter section captures the immigration matter type (Express Entry, Spousal Sponsorship Overseas, Study Permit, PFL Response, and the rest of the canonical list — the same enum the SA builder uses), the matter code (IRCC's program code where applicable), and a free-text description of the specific factual context. The scope section captures the included and excluded items lists. Both lists are rich-text-editable; the rendered clause uses bullet points for each item. The validator emits a warning when scope is empty (you can ship without scope items, but a clean co-counselling agreement should describe what each RCIC is doing).

Service components are the itemized breakdown of work each RCIC is responsible for. Each row carries: a label, a dollar amount (the portion of the total fee allocated to this component), a responsible party (lead RCIC, invited RCIC, or both — defaults to the lead RCIC on add for mechanical-blocker auto-clearance), an applicant kind (main client, family member, sponsor, other), and an optional familyMemberId pointer that links the component to a specific family-member row. The components sum auto-validates against the total professional fee in the fee arrangement section; mismatches emit a validator blocker. The Auto-sync button (right rail) re-calculates the fee arrangement totals from the component breakdown when you want the components to drive the totals rather than the other way around.

Role matrix

The role matrix captures who is responsible for what at the matter level (as distinct from the per-component allocation in service components). Each row is a free-text role label (Overall matter responsibility, Client communication, Application drafting, IRCC liaison, File retention, etc.) plus a dropdown that picks the responsible party (lead RCIC, invited RCIC, or both). When you populate scope and the matrix is empty, the platform auto-seeds a default Overall matter responsibility row assigned to the lead RCIC — this is the mechanical-blocker auto-clearance pattern that keeps the validator quiet without forcing you to type the row by hand. You can edit or delete the auto-seeded row freely.

The validator

The validator runs on every autosave and surfaces blockers and warnings in the sticky right rail. Blockers prevent send; warnings are advisory. Common blockers: missing client name or email, missing invited-RCIC identity, parties' fee shares do not sum to the total, service-component fees do not match the fee-arrangement total, any locked clause body has been deleted (you cannot delete locked clauses; the editor disables the delete button on them, but the validator catches any attempt to bypass that gate), the client-consent affirmation timestamp is missing (this should never trigger because the new-CCA form requires it before insert, but the validator carries the rule as defense in depth).

Mechanical-blocker auto-clearance is a deliberate UX choice. Several blockers fire from missing values that have a sensible default: the responsible party on a freshly-added service component (defaults to lead RCIC), the familyMemberId link on a component whose label matches an existing family-member row (auto-resolved by name match, parentheticals stripped), the role-matrix row when scope is populated but the matrix is empty (auto-seeded with an Overall row), the fee arrangement notes when the dollar split is non-trivial (auto-composed from the share amounts if the notes field is empty). These auto-clearances run on save and are idempotent — re-running them does not overwrite your manual edits, only fills empty fields.

Templates shared with Service Agreements

CCA does not have its own template library. The Apply template action in the CCA builder reads directly from the Service Agreement template library at /dashboard/agreements/settings/templates — the same library you use to publish tenant and global templates for client-facing Service Agreements. The picker modal opens with a search box (matches against template name, matter code, and service category), a What's the difference? help section that explains the three apply modes (fill empty, replace, add component), and three tabs: All (n), Your templates (n), and Global (n), each with its own count. Tenant templates sort ahead of globals in the All view. Each row carries a maroon Your template pill or a charcoal Global pill so you can tell at a glance who owns the template.

When you apply a template, the platform maps the SA-shaped template payload onto the CCA draft. Scope, milestones (which become the joint-counsel handoff milestones in the CCA), payment schedule (which informs the fee arrangement rather than living as a separate schedule), indicative timeline, included family members, and the recommended service components all flow through. The chosen-fee MoneyInput in the picker lets you pre-set the total fee at the moment of apply; the components scale proportionally if they are priced as percentages or stay at their template-default amounts if they are priced absolutely. After apply, your manual edits in the builder take precedence; the next autosave pass writes the merged draft to the database.

> **Note**: The shared library means publishing a new template (or amending an existing one) updates both the SA builder and the CCA builder at the same time, with no migration or sync step. Equally, when you create a tenant template for a niche matter your firm specializes in, that template becomes available in your CCA builder too — useful when you and a colleague co-counsel on a matter type you have already templated for your own client work.

AI Populate

AI Populate in the CCA builder reuses the same Claude Sonnet 4.6 extractor that powers Service Agreement AI Populate. Click the AI Populate button in the right rail, drag-and-drop one or more documents (passport bio page, refusal letter, prior agreement between the two RCICs, intake summary, anything that carries facts the AI can extract), pick a per-upload document role (passport, refusal letter, prior agreement, other), optionally type a tenant prompt that adds context for the AI, and click Run. The extraction takes thirty to sixty seconds on a typical multi-document set; an indeterminate progress overlay covers the modal.

The extractor returns SA-shaped proposals: client identity fields (full name, email, phone, address, date of birth, country of birth, citizenship), matter framing (matter type, matter code, scope items), recommended template with a chosen fee, recommended service components, and recommended family members. The CCA apply step maps each SA-shaped path onto the CCA draft. Paths that have no CCA equivalent (parties.coCounsel.* — the CCA's invited RCIC is captured separately; parties.thirdPartyPayer.* — CCA does not use a third-party payer; parties.designatedPerson.* — CCA does not use a designated person; matter.complexity — CCA does not track complexity; client.business* — CCA does not track business client fields) are silently dropped on the client side so the row list shown to you only includes proposals that will actually apply. The modal shows per-row Accept and Discard buttons; the recommended template gets its own card with an Apply button that calls the apply-template route with mode=add-component, which is the path that auto-fills the recommended service components into the existing draft without overwriting your manual edits.

> **Note**: AI Populate counts toward your tenant's daily AI usage quota — the same shared 50-per-day pool that the in-dashboard AI Assistant, the booking-page AI document summary, and the Service Agreement AI Populate all draw from. If you have exhausted the day's quota, the modal returns a clear error and the run does not consume a quota slot. Premium tenants get the larger quota; Basic tenants do not see the AI Populate button (Basic does not unlock CCA at all, so this is academic).

Preview HTML and PDF

The right rail carries two preview buttons. Preview HTML opens a server-rendered HTML view of the agreement as the invited RCIC will see it on the portal: the full document, with the 22 clauses ordered as the renderer composes them, with substitution-map values filled in from your current draft state. Preview PDF opens a fresh PDF render with a large DRAFT watermark across every page and a multi-line audit footer at the bottom (DRAFT - not yet executed - generated YYYY-MM-DD HH:MM:SS UTC). Both previews live-update on every autosave so what you see is what the next save would render.

The PDF watermark is removed at finalization. The signed PDF that lands in the cca-pdfs storage bucket after both RCICs sign carries the multi-line audit footer (signer name and role, redacted IP /24, token-hash prefix, UTC stamp) on every page but no DRAFT watermark. The signed PDF is owner-password protected through pdf-lib's encryption: anyone with the PDF can open and read it, but modifying, annotating, filling forms, or assembling pages out of it is blocked at the PDF-reader layer. Print and copy and accessibility are allowed.

Send for signature

When the validator is clean, the Send for signature button in the right rail unlocks. Clicking it does several things atomically: writes the cca_parties rows (one for the initiating RCIC, one for the invited RCIC) with their identity snapshot at the moment of send, mints a portal token for the invited RCIC (32-byte CSPRNG, hashed before

persisting — plaintext never lives in the database), sends the invitation email to the invited RCIC's address with the tokened portal link, and flips the matter status from Draft (or Ready to send) to Sent for review.

The invitation email is bilingual (EN and fr-CA stacked top to bottom in the same message body) and branded as RCIC App via Investatech, with reply-to set to the initiating RCIC's professional email so a quick reply lands in your inbox. The subject line names the matter (CCA reference plus client name). The body explains what a CCA is, identifies you as the requester, names the client, and links to the portal. Once Sent for review, the only edits you can make from the dashboard are: resend the invitation (rotates the token; the prior link no longer works), cancel the matter (with a recorded reason), or wait for the invited RCIC to act.

The invited RCIC's portal

The invited RCIC clicks the tokened portal link in their invitation email and lands on /co-counselling-agreement/<token>. The page is gated by an email confirmation step (they confirm their email matches what is on file; no OTP is sent for this — the token in the URL is the capability, the email confirmation is a soft second factor to prevent forwarded-link impersonation). On confirm, they see three views in tabs: Sign (the rendered agreement plus a typed-signature box and a Confirm and sign button), Decline (a structured-reason dropdown plus a free-text notes box and a Decline button), and Request changes (a free-text notes box describing what changes they want and a Submit request button).

Sign produces the invited RCIC's signature on the cca_parties row (signature method: typed; typed name: the value they entered; signed_at: now) and emits a signed audit event. The matter status stays Sent for review until you (the initiating RCIC) countersign — at that point tryFinalizeCca runs the completion pipeline (see next section). Decline produces a declined timestamp and reason on their party row and flips status to Declined; you receive a notification email and can re-open the draft, change terms, and resend (to the same or a different invited RCIC). Request changes produces a change-request payload on their party row and flips status to Changes requested; the lifecycle view shows the request notes prominently with a Re-open in builder button that flips status back to

Draft so you can edit. When you (the invited RCIC's tenant counterpart) are an RCIC App tenant the portal is replaced by an in-dashboard surface and the email confirmation step is skipped (you are already authenticated in your tenant session).

Countersign and finalize

When the invited RCIC has signed, the matter detail page on your dashboard shows a Countersign button next to the invited RCIC's signed-at timestamp. Clicking opens a small modal that captures your typed name (defaults to your tenant profile display name) and a Confirm and counter-sign button. On confirm, the platform writes your signed-at timestamp on the initiating-RCIC party row and runs tryFinalizeCca: a SQL-guarded UPDATE flips the matter status from Sent for review to Completed (the guard ensures only one finalize wins if two browser tabs race), renders the signed PDF, encrypts the PDF buffer under your tenant DEK, uploads to cca-pdfs/<companyId>/<ccaId>/signed-<timestamp>.pdf.enc, attaches the plaintext PDF to two completion emails (one to you, one to the invited RCIC), and emits a finalized audit event.

Both completion emails carry the signed PDF as an attachment, not just a portal link. The signed PDF is the canonical record of the executed instrument; once attached and sent, the email is the durable copy each RCIC can save in their own client-matter file. The email also includes a portal link in case the attachment is stripped by an aggressive mail filter on the recipient's side. The PDF in the cca-pdfs bucket stays encrypted at rest under your tenant DEK; only you (the owning tenant) can decrypt it via the dashboard. The invited RCIC retains their copy through the completion email and, when they are an RCIC App tenant, through a read-only view in their own dashboard.

Auto-generated client Service Agreement

When you are the lead RCIC (you picked 'We are the lead' on the new-CCA form) and the CCA reaches Completed, the platform automatically runs exportCcaToSa. The export builds a partial SaDraftData from the

CCA: client identity (full name, email, phone, address, date of birth, country of birth, citizenship — decrypted from the CCA on the export side, re-encrypted on the SA insert side under the same tenant DEK), matter framing (type, code, scope), family members, service components (mapped through the shared component shape), and the invited RCIC pre-loaded into the SA's co-counsel party slot with the fee share carried across from the CCA's fee arrangement section. The platform inserts a fresh service_agreements row with a source_cca_id pointer back to the CCA, stamps the CCA's generated_sa_id field with the new SA id, and flips the CCA status from Completed to Service Agreement draft created.

The CCA lifecycle view now shows a maroon Open the generated Service Agreement link that deep-links to the SA detail page. The SA is a draft — you finish whatever scope, payment-schedule, or per-clause edits are needed for the client-facing instrument, validate, finalize, send to the client, and proceed through the standard SA signing lifecycle. The co-counsel block in the SA carries the invited RCIC's identity and the fee share, so the client signs a document that names both consultants and binds them to the arrangement they already executed in the CCA.

The export is idempotent: the guard generated_sa_id IS NULL means re-running exportCcaToSa on a CCA that has already exported does nothing (returns the existing SA id). The CCA row keeps its source_cca_id pointer for life — even when the CCA itself is later soft-deleted, the SA's pointer survives so the audit chain from client SA back to the CCA back to the co-counselling negotiation is preserved.

> **Note**: When you are not the lead RCIC, no SA is auto-generated on your side. The lead RCIC's tenant (whether on the platform or external) is responsible for producing the client Service Agreement. When the lead is an RCIC App tenant, the platform runs the same exportCcaToSa logic on their side; when the lead is external, the lead RCIC produces the SA in whatever tool they use, and you receive a copy out of band.

CCA Settings (shared with Service Agreements)

The CCA Settings entry in the sidebar lands you on a deliberately small settings hub at /dashboard/co-counselling/settings. The page is Owner-only (Admin and team members get redirected to /dashboard) and currently carries one card: Templates (shared with Service Agreements), with a one-line explanation that CCAs pull scope, milestones, payment schedule, and the typical professional fee from the same template library you use for Service Agreements, and an Open Service Agreement templates button that links to /dashboard/agreements/settings/templates.

The deliberate minimalism reflects the architectural choice that CCA reuses SA infrastructure rather than maintaining a parallel one. There is no per-CCA template, no separate template library, no separate practice-defaults sheet, no separate fee profile. Everything that governs the substance of an SA (templates, clause defaults, fee profile, billing-on-signing global toggle) governs CCA in lockstep. When more CCA-specific settings ship in a future phase (per-tenant defaults for fee-share patterns, default invited-RCIC contact, etc.), they will live on this same settings page and not bleed into the SA settings sheet.

> **Note**: Because templates are shared, any edit to your tenant template library updates both modules at the same time. This means you maintain one library, not two, and the operational cost of running both modules is the cost of running one. Equally, when CICC updates a clause requirement in a way that affects your templates, you fix it once and both modules pick up the fix on the next save.

Reminders and the daily cron

A daily cron at /api/cron/cca-reminders runs at 13:35 UTC and sweeps the matter table for two reminder windows: matters in Waiting for profile-share consent for more than three days (one reminder; further reminders are operator-handled) and matters in Sent for review for more than seven days (one reminder per week, capped at three reminders before the platform stops nudging). Each reminder rotates the token on the affected party row

(the prior link no longer works), sends a fresh email with the new link, and writes an audit-ledger entry for the reminder.

The reminder cadence is deliberately conservative for a peer-to-peer instrument. Service Agreements that sit too long with a client get aggressive reminders; CCAs that sit too long with another RCIC get a single nudge and then leave the relationship to operate out of band. If you find your invited RCIC consistently unresponsive past one reminder, the right move is usually a phone call rather than a third email — and the platform is built to reflect that.

Encryption, storage, and the audit ledger

Every sensitive value in the CCA module is encrypted at rest under your tenant data-encryption key (DEK), wrapped under the master key-encryption key (KEK) that lives in Vercel environment variables. Encrypted columns include the client name and email, the invited RCIC name and email (for external invitations; the tenant-path invitations read from the verified shared profile), the fee arrangement notes, the matter description, the scope items, the role-matrix free-text labels, and the change-request notes. Email columns also carry blind-index hash columns so exact-match lookups (find this CCA by client email) work without ever decrypting the canonical value.

The signed PDF lands in the dedicated cca-pdfs storage bucket (created by migration 173). The bucket is private with no public read policy. The PDF buffer is encrypted with the tenant DEK before upload, and the storage object's content-type is forced to application/octet-stream so a leaked signed URL does not let a browser auto-render the PDF. Decryption happens server-side on a per-request basis when an authorized actor downloads through the dashboard. Only you (the owning tenant) and the platform service role can decrypt; the invited RCIC's plaintext copy lives only in the completion email attachment they received at finalization.

The audit ledger is the cca_events table. Every state-changing operation writes an INSERT-only row: created, profile_share_requested, profile_share_granted, profile_share_declined, draft_edited (one row per autosave batch

— coarse granularity to avoid the ledger blowing up on continuous editing), validator_passed, sent_for_signature, party_signed, party_declined, change_requested, countersigned, finalized, sa_draft_created, reminder_sent, cancelled. Each row carries timestamp, actor (your user id or the invited RCIC's identifier or the cron sentinel), IP redacted to /24, and a JSON payload with the structured details. A Postgres trigger raises on any UPDATE attempt against cca_events so the ledger is tamper-evident even from the service role.

Integrations with other modules

CCA shares more architecture with Service Agreements than with any other module. The shared template library, the shared Claude AI extractor, the matching clause vocabulary, the parallel builder UX, and the auto-generated SA at completion all mean a tenant who knows the SA module knows the CCA module already. The relationship is bidirectional: every SA generated by a CCA carries a source_cca_id pointer back to the CCA; the SA detail page renders a back-link card when source_cca_id is set; the CCA lifecycle view renders a forward-link to the SA when generated_sa_id is set. The two records are linked for life.

- Service Agreements: the closest integration. CCA reuses SA's template library, AI Populate extractor, clause render pipeline, and builder shape. When you are the lead RCIC, an SA is auto-generated at CCA completion with the invited RCIC pre-loaded as co-counsel.
- Single Bills: when the auto-generated SA reaches fully-signed, the global Bill on signing setting drives auto-bill creation. The CCA is not directly billed — it is a peer-to-peer instrument between two RCICs, not a client-payable retainer — but the downstream client SA is, just like any other SA.
- Transfer Room: the auto-generated SA's fully-signed event provisions a Transfer Room for the client matter; the co-counsel block in the SA carries the invited RCIC, so the Transfer Room's participant list can name both consultants at activation time.
-

AI Support: CCA's AI Populate counts toward the shared daily AI quota; the in-dashboard AI Assistant can answer questions about your CCA workflows just like it does for SA workflows (same context, same model).

- Active File Review: when a matter starts as an AFR and grows into a multi-RCIC engagement, you can spawn a CCA from the AFR-generated SA's co-counsel block. The two modules thread the same client identity throughout, so you do not re-key client name, email, or matter context at any handoff.

If you operate a multi-RCIC practice and find that most of your matters need a CCA at some point, the practical workflow is: maintain your template library on the SA side once (it powers both modules), use the AFR or Intake Forms or Bookings modules to acquire matters, generate CCAs from the dashboard when a co-counselling arrangement is needed, let the platform auto-generate the client SA at CCA completion, and let the SA's fully-signed event provision the Transfer Room. Five modules, one client identity, no re-keying — that is the design intent.