AUTOMATION INTEGRATOR

AI · CONNECTING TO YOUR BUSINESS

A PRACTICAL PLAYBOOK

From Zero to an AI-Run Trade Business

How one commercial cabling contractor put AI to work across sales, estimating, finance, project delivery and marketing, and the exact order to do it in, for any trade or services business.

Kieron Henare
Automation Integrator

August 2026

CONTENTS

What's inside

01 Who this is for, and the one idea that matters most 3
The single design rule that makes this safe to trust

02 The mental model, five operating principles 4
The reusable patterns behind every automation in this document

03 The toolchain, in plain language 5
Six categories of tool, plus how Skills and MCP connectors actually build the automations

04 What got built, by business function 7
Communications, sales, estimating, finance, CRM, project delivery, marketing, oversight

05 The infrastructure layer 11
How it actually runs, and the mistake worth learning from

06 Real costs 12
A grounded monthly figure, not a guess

07 The migration skeleton 13
A phased plan for doing this in a different trade business

08 Mistakes made and corrected 15
The honest retrospective

09 A short glossary 16
For anyone starting genuinely from zero

PART 01

Who this is for, and the one idea that matters most

If you run a trade or services business, you are almost certainly doing five jobs at once: sales, estimating, project delivery, finance, and marketing, usually before 7am and after the crew's gone home. AI doesn't remove any of those jobs. What it removes is the *mechanical* work inside each one: reading the same email three times, retyping a number from a PDF into a spreadsheet, chasing the same overdue invoice, remembering which tender closes Thursday.

The single idea that made this whole build trustworthy, and the one to hold onto above everything else in this document:

> **AI does the mechanical work and shows its working. A human makes the judgment call and presses send.**

Every automation described below follows that rule in a concrete, checkable way, not as a slogan, but as an actual design constraint: drafts instead of sent emails, marked-up PDFs instead of bare numbers, a blank "To" field instead of an address, a confirmation prompt before anything financial. That discipline is *why* it was safe to build this much, this fast, without anything going wrong that mattered. If you take one thing from this document and ignore the rest, take that.

PART 02

The mental model, five operating principles

These aren't abstract values, they're the specific patterns that show up in nearly every automation built here. Recognize them; they're the reusable part.

1 Draft, never send.
Anything that goes to a customer, supplier, or creditor is written by AI and reviewed by a human before it leaves. Technically enforced, not just requested, e.g. the debtor-chase automation opens its draft emails with the "To" field physically blank, so there is no code path that can send it by accident.

2 Show the working, not just the answer.
A plan takeoff produces a marked-up PDF with every counted symbol circled and numbered, never just "142 outlets." A pricing automation produces the filled spreadsheet, not just a total. If a human can't quickly verify the output against the source, the automation isn't finished.

3 Read-only until proven, then advisory, then (rarely) autonomous.
Every financial automation in this build is explicitly "advisory only" or "draft only", it reads the accounting package, computes something, and emails a person. None of them move money. The one automation that *does* write records automatically (logging sent proposals into the CRM) was only granted that because a wrong CRM entry is cheap to fix and doesn't touch cash.

4 Benchmark before you trust it.
Before the plan-counting tool was used on a single real tender, it was run blind against 11 already-priced historical jobs and the results compared against what a human had actually counted and won work on. Only after that matched closely enough did it go live. Don't skip this step for anything that touches money or a customer relationship.

5 Verify, don't extrapolate.
If a summary says something is "done," open the actual file and look, every time. Automations should behave the same way about their own outputs, a status report is never a substitute for checking the real state of the world.

PART 03

The toolchain, in plain language

You don't need twenty tools. You need roughly six categories, and probably one tool per category. Here's what each layer actually does and why it's there, using what CCSI ended up with as the worked example.

Layer	What it does	CCSI's choice	Roughly why
The AI itself	Reads, writes, reasons, and calls the other tools	Claude (Claude Code + the Claude desktop app)	Handles both "sit with me while I work" and "run unattended on a schedule" from one vendor
Business data & communication	Email, calendar, files, contacts	Microsoft 365 Business Premium + SharePoint/OneDrive	Most trade businesses already have M365 or Google Workspace, this is the layer AI needs read/write access to
Accounting	Invoices, bills, cash position, aged debt	Xero	Has a real API/connector, matters more than which package is "best"
Automation scheduling	What actually fires the AI on a timer	Windows Task Scheduler	Free, native to the machine, doesn't depend on any app staying open
Custom web tools	A crew-facing app, a public website	Azure (App Service + Static Web Apps)	Talks natively to business data already in Microsoft's ecosystem
Always-on dashboards / small agents	Small things that need to run 24/7 cheaply	Cloudflare Workers	Effectively free at this scale, nothing to manage
Marketing distribution	Scheduling and publishing social content	Buffer	Real API, ~$6/channel/month
Remote access	Checking on the machine running everything, from anywhere	Tailscale + Windows Remote Desktop	Private network, nothing exposed to the internet
Industry-specific portals	Wherever your leads/tenders actually live	BTSL (NZ construction tenders)	The one genuinely trade-specific row, swap for your industry's equivalent

The pattern to copy, not the specific vendors: one AI layer that can both work alongside you and run unattended, wired into the business software you already use, with a free/cheap scheduler triggering it, and a place to build anything custom off-the-shelf tools can't do.

How this actually gets built: Skills and connectors

Two more pieces make the "AI itself" row work in practice, worth understanding if you're going to build this rather than just buy it.

SKILLS

The instructions for each specific job

Every automation in this document isn't one big program, it's a separate, self-contained "Skill": a plain-text instruction file that tells the AI exactly how to do that one job, written the way you'd brief a new employee, what to check, what the rules are, what "done" looks like, what to never do. The debtor-chase Skill doesn't know anything about email drafting; the email-drafting Skill doesn't know anything about takeoff counting. Each one is small, readable, and editable on its own, which is what makes 30+ automations manageable instead of one unreadable mess.

> **Why it matters:** when a Skill gets something wrong, you fix the specific instruction that caused it, in plain English, not code. That's what let hard-won lessons (a labour-rounding rule, a pricing-safety check, a folder-naming convention) get written down once and never re-litigated.

MCP CONNECTORS

The actual access

Skills tell the AI what to do; MCP (Model Context Protocol) connectors give it the ability to actually do it, a real, permissioned bridge into each piece of business software: read this calendar, draft this email, look up this invoice, update this CRM record. Without a connector, the AI can only talk about your business in the abstract. With one, it can read the real inbox, the real accounting package, the real project files, and act inside them, within whatever limits you set.

> **Why it matters:** this is the layer that turns "an AI that gives good advice" into "an AI that does the work." It's also the layer worth being most deliberate about, a connector with write access is a real capability, and the draft-not-send discipline from Part 02 exists specifically to keep that capability safe.

PART 04

What got built, by business function

The actual inventory. Each entry: the problem it solves, how it works, and the human checkpoint built into it.

4.1 - COMMUNICATIONS

The front door

Problem: A trade business owner's inbox mixes customers, suppliers, crew and noise, every reply meant re-reading the whole thread.

Built: An automation drafts a reply to every new email (never sends), matched to the sender's context, pulling the right price, the right file, the right tone. Runs every 15-30 minutes on weekdays. A separate triage pass sorts flagged/to-do emails into actioned/archived/converted-to-task.

> **Checkpoint:** every draft sits in the inbox's Drafts folder for review. Replaced a paid third-party drafting tool once trust was earned, the deciding moment was watching it correctly pull a customer-safe *sell* price while refusing to leak the *cost* price sitting in the same file.

Generalize it: the single highest-value, lowest-risk automation to build first in any trade business, a bad draft costs ten seconds to delete; a bad email costs a customer relationship.

4.2 - SALES PIPELINE / LEAD INTAKE

Problem: New tenders and quote requests need triaging, filing, and calendaring before anyone can act on them.

Built: Daily scan of the inbox and every relevant subfolder for new tender/pricing requests, filing attached plans into the right customer folder and calendaring the closing date. A twice-weekly check of the industry tender portal decides GREEN / RED / GREY on relevance and books calendar time for the greens. A website "instant estimate" tool lets a prospect self-serve a rough price range, capturing a lead with zero human involvement. Once a job is actually won, a daily scan raises it in the job-numbering register, pipeline tracker, and folder structure automatically, so the admin of starting a job never lags behind the win. A weekly scan of main-contractor award notices flags the ones relevant to the business and files them against the right customer record.

> **Checkpoint:** GREY calls are always a human decision. Nothing is quoted or committed at this layer, it only sorts and files.

Generalize it: every trade has some version of "a new work request arrives and needs triaging before pricing can start," worth automating even without a public tender portal.

4.3 - ESTIMATING AND QUOTING

The highest-skill layer, AI assists, doesn't decide

Problem: Counting items off a drawing and building a priced quote is slow and error-prone under deadline pressure, and it's where real judgment and margin live.

Built: A plan-counting tool reads a legend, learns that drawing set's specific symbols, counts every match across every sheet, and produces a marked-up proof PDF plus a filled pricing-sheet template. A variation-pricing tool turns a "please price this" request into a filled sheet and a drafted reply, following a formula-driven template so the spreadsheet's logic is never hand-typed over. A proposal-generation tool is triggered on request, not on a timer, pricing is a judgment act.

> **Checkpoint:** every count gets a visual proof document, benchmarked against 11 real historical jobs before touching a live tender. Deliberately the most human-involved layer of anything built, it's where the money decisions live.

Generalize it: whatever your trade's "count it off the drawing and price it" step is, budget real testing time, this took roughly ten iterative rounds against a human counting the same job by hand before it was reliable on its hardest case.

4.4 - FINANCIAL ADMIN

The layer with the strictest rules

Problem: Debtor chasing, cashflow forecasting, payment runs and reconciliation are essential and the first things to slide when busy.

Built (all wired to the accounting package, all "advisory or draft, never autonomous"): **Debtor chase**, twice-monthly, drafts one reminder per overdue customer, recipient field deliberately blank. **Cashflow projection**, fortnightly 30/60/90-day forecast, emailed as a report. **Payment run prep**, monthly, checks bills due against available cash, emails an approval-ready schedule; never pays. **Supplier statement reconciliation**, monthly, flags mismatches before payment. **Progress-billing invoice drafting**, on demand, raises the gap between claimed and invoiced as a draft invoice only. **Project-level reconciliation**, on demand, checks one job's actual costs against budget. **Purchase order raising**, on demand off a won job, creates the PO in the accounting package, files the PDF, and drafts the email to the contractor. **Contractor invoice allocation**, monthly, files every contractor claim into its job folder and rolls the payment schedule. **Supplier invoice processing**, daily, files the invoice into the right job folder and drafts the matching bill. **Variation-billing audit**, monthly, checks every completed variation actually got raised as a customer invoice, not just priced and left sitting in a folder.

> **Checkpoint:** every one of these produces a draft, an advisory email, or requires a manual trigger with an explicit confirmation step. None move money autonomously, deliberately, never crossed.

Generalize it: if your accounting package has an API (most do), this whole category is copyable almost as-is, the reports change, the "advisory only" discipline doesn't.

4.5 - CUSTOMER RELATIONSHIP TRACKING

Problem: Every priced proposal should become a tracked opportunity, but logging CRM entries after every quote is exactly the task that gets skipped when busy.

Built: A daily scan of sent mail for outbound proposals/quotes, extracting recipient and value, creating or updating the matching CRM contact/company/deal automatically. The weekly main-contractor award scan feeds the same CRM, moving a customer's stage forward automatically when they're reported as awarded work.

> **Checkpoint:** the one automation allowed to write without a human approving each write, deliberately, because a wrong CRM entry is cheap to fix (unlike a wrong email or invoice), and never inventing a value it can't read with confidence.

Generalize it: a good template for deciding which automations can act unsupervised, ask what a mistake actually costs, not just whether one is possible.

4.6 - PROJECT DELIVERY / ON-SITE QA

Problem: Getting consistent quality-assurance sign-off from crews on site, with photo evidence filed against the right job, normally doesn't happen or happens on paper that gets lost.

Built: A purpose-built web app, office side (company login) is a project dashboard with document management and QA tracking; crew side needs **no login at all**, just a link or QR code per project, because requiring crew logins was the actual adoption blocker. Crew tick off checklists with mandatory photos, sign, submit; a supervisor approves or declines with a reason; only approved inspections file into the company's document system.

> **Checkpoint:** nothing files until a supervisor explicitly approves it. Crew-facing views are stripped of all pricing information, enforced in code, not just policy.

Generalize it: a link/QR-code front door with no account requirement is very often the real unlock for on-site adoption, in any trade.

4.7 - MARKETING

Problem: Consistent content and outreach take time that's the first thing dropped when busy, exactly when it matters most.

Built: Brand positioning locked once, then a weekly automation builds and schedules the next week's social content across every channel, plus a performance report after every posting day. Self-serve lead-generation tools added to the website (instant price-range estimator, photo-based quote request) as low-friction entry points. A cold-outreach engine is scoped as the next phase.

> **Checkpoint:** approve-first by explicit design, no autonomous posting or spend until the content engine has earned that.

Generalize it: "lock the positioning once, generate content against it on a schedule" is fully reusable, your message will differ, the mechanism won't.

4.8 - OVERSIGHT

Watching the watchers

Problem: Past a handful of automations, "did everything actually run today?" becomes its own job.

Built: A dashboard showing every automation's live status in one place, deliberately framed as a team roster rather than a technical report, because that's a genuinely more useful mental model once you're past a handful of automations. A daily health-check emails a flagged alert if anything's broken. Two things were added once the fleet grew past a couple of dozen automations: a message bus that lets automations hand work directly to each other instead of each one polling independently, and a nightly review that reads back through the day's work and proposes new standing instructions or fixes for approval, so the operating knowledge compounds instead of living only in one person's head.

Generalize it: don't skip this layer past 5-10 automations, "I'll remember to check" stops working, and you need something that flags when it *hasn't* run. Chaining and self-review are a later-stage layer on top, worth adding once polling-based scheduling starts to feel laggy rather than from day one.

PART 05

The infrastructure layer

The two-attempt story, worth telling honestly: the first attempt at scheduling these automations used the AI tool's own built-in cloud scheduler and desktop-app scheduler. This mostly worked, but desktop-scheduled tasks only run while the app is open, and the cloud scheduler occasionally missed a fire with no obvious cause. Everything was later migrated onto the operating system's own native task scheduler, calling the AI tool directly. This decouples automations from any app needing to be open, the trade-off is it now depends on the physical machine being powered on and signed in.

What actually keeps a large automation fleet running on one physical laptop

- Task Scheduler entries set to run only while a specific user is logged in, a platform restriction on business-managed devices; "run whether logged on or not" isn't available without extra setup.
- The one thing that's free and worth doing on day one: set the machine's power plan so it never sleeps while plugged in. Confirmed on this build: a laptop can stay signed in (even locked) for days straight and every scheduled task keeps firing.
- What that setting doesn't solve: a machine fully shut down, signed all the way out, or forced to restart by an OS update. There's no scheduler-side fix, it's inherent to "must be logged on" scheduling.

The redundancy option, tried, and reversed

If a single physical machine running your fleet is a risk worth closing (theft, hardware failure, needing to travel without pausing everything), a second always-on machine in the cloud is the obvious fix, and it's worth knowing this build actually tried it before recommending it:

1. A managed cloud PC service was set up as the second machine, on the reasoning that a hosted desktop would be simpler to keep in sync than a raw virtual machine.
2. It failed the one requirement that mattered: it deallocated on every disconnect, so scheduled tasks stopped firing the moment nobody was actively connected, which defeats the entire point of an always-on redundant machine.
3. The migration was reversed. The fleet runs from the original physical machine today, with no fallback, a known and currently unresolved single point of failure, not a solved problem dressed up as one.
4. If you try this yourself, confirm the specific product's actual power-state behaviour when disconnected before migrating anything to it, that one property is the whole decision, not the sticker price or the desktop-app licensing.

PART 06

Real costs

Grounded in actual current pricing where confirmed; a couple of rows are estimates, flagged as such. All NZD unless noted, approximate.

Item	Cost	Notes
AI subscription (top usage tier)	~US$100-200/mo	Lower tiers exist for lighter use
Microsoft 365 Business Premium	Check your existing invoice	Most trade businesses already have some M365/Workspace plan, the win is using what you already pay for
Xero	NZ$31-84/mo (excl. GST)	Any accounting package with a real API works the same way
Buffer (social scheduling)	~$6/channel/mo	5 channels ≈ $30/mo
Azure App Service (custom web app)	~NZ$20-25/mo	Estimate, get a live quote for your spec
Azure Static Web App (website)	Free tier usually sufficient	For a simple site
Cloudflare Workers (dashboards)	Free tier usually sufficient	At this scale
Tailscale (remote access)	Free tier sufficient	For personal/small use
Cloud PC (redundancy)	~NZ$95/mo	Tried and reversed here, see Part 05, budget for it only once you've confirmed the specific product stays running while disconnected

The honest bottom line: none of this required a large budget to start. Email-drafting and financial-admin cost nothing beyond a normal M365 + accounting + AI subscription. The single most expensive line is the AI subscription itself, everything else is closer to $20-90/month per tool, redundancy remains an unsolved cost line here, not a solved one.

PART 07

The migration skeleton

If you're starting from zero, this is the order that actually worked, not because it's abstractly "correct," but because each phase built the trust and infrastructure the next phase needed.

PHASE 0 · WEEK 1

Foundation

Get an AI subscription that can both work interactively with you and run unattended on a schedule. Make sure you (or whoever's driving this) has admin rights on your business email/file system. Pick **one** person as the operator who reviews drafts and approves changes.

PHASE 1 · WEEKS 1-2

Communications

Start with draft-only email replies, the safest possible starting point, and it teaches both you and the AI the shape of your business's real correspondence. Add inbox triage once drafting feels trustworthy.

PHASE 2 · WEEKS 2-6 (THE SLOW ONE)

Your trade's core estimating/pricing workflow

Identify the single most repetitive, time-consuming step in turning a lead into a priced quote. Build it to produce visible proof of its work, never a bare answer. Budget real benchmark time against your own completed jobs before trusting it live, this was the longest phase in this build and is worth not rushing.

PHASE 3 · WEEKS 3-5 (PARALLEL TO PHASE 2)

Financial admin

Connect your accounting package. Start with read-only reporting before anything that drafts or creates records. Add draft-only debtor chasing and invoice drafting once the reporting numbers are trusted.

PHASE 4 · MONTH 2+ (IF YOUR TRADE NEEDS IT)

Customer/crew-facing tools

Worth a custom build only after Phases 1-3 are solid, it's the most technically involved piece. The single most important design decision: don't require field staff to have logins if you can avoid it.

PHASE 5 · MONTH 2+ (PARALLEL)

Marketing

Lock your core positioning in writing before generating any content against it. Start approve-first. Earn autonomy over time, don't grant it upfront.

PHASE 6 · ONCE 5+ AUTOMATIONS EXIST

Infrastructure hardening

Move off any app-dependent scheduler onto your operating system's native one. Set your always-on machine to never sleep while powered. Build a simple oversight view, even a single status email is enough at first.

PHASE 7 · ONLY ONCE PROVEN RELIABLE, STILL UNSOLVED HERE

Redundancy

A second always-on machine closes off the single point of failure (theft, hardware failure, needing to travel without pausing everything), worth planning for once your setup is proven. Test the specific product's power state on disconnect before you commit, a managed cloud PC that deallocates when nobody's actively connected fails the one job it exists to do, see Part 05.

PART 08

Mistakes made and corrected

Worth including because these are the actual lessons, not the sanitized version.

> **An earlier attempt to migrate scheduling to code failed silently**, a first pass only produced a mapping document, not a working migration. The successful version took real hands-on time, not a single automated pass. Don't expect an infrastructure migration like this to be a one-shot job.

> **A browser-upload-dependent workflow was ported, tested, and deliberately rolled back**, the new platform's browser-automation tooling couldn't reach local folders the old one could. Test the specific technical capability a workflow depends on before assuming a migration will carry it over, and be willing to roll back rather than force a broken tool to "mostly work."

> **The cloud scheduler had a real, unexplained failure mode**, every scheduled automation on one cloud platform silently stopped firing account-wide for a full day, with no status-page incident and no consistent fix. A known, recurring platform-level risk, worth knowing before depending entirely on any single cloud scheduler for anything time-sensitive.

> **A licensing assumption was wrong until checked**, extra software cost was nearly budgeted for a redundancy project before checking the existing business subscription, which already included the needed rights. Check what you already pay for before assuming you need to buy more.

> **A "missing" automation turned out not to be missing**, it existed, just as an on-demand tool rather than a scheduled one. Before concluding something isn't built, check both the scheduled list and the on-demand tools.

> **The redundancy migration itself had to be reversed**, a managed cloud PC set up as the second always-on machine deallocated every time it disconnected, silently killing every scheduled task on it. It looked proven in testing while someone was actively connected and failed the actual job the moment nobody was watching, which is exactly when an unattended scheduler needs to keep running. Migrated back to the original machine with no fallback in place, still an open problem. Test disconnected behaviour specifically, not just whether the thing boots and runs while you're logged in.

PART 09

A short glossary

For anyone starting genuinely from zero.

AI agent / automation

A scripted task that tells the AI what to do, on a schedule, without a human typing the request each time.

Scheduler

Whatever actually triggers the automation at the right time, built into the AI tool, or (more reliably) your computer's own operating system scheduler.

Connector

The technical bridge that lets the AI actually read and write to a real piece of business software, rather than just talking about it in the abstract.

Draft vs. send

The most important distinction in this document. A draft sits waiting for a human. A send is irreversible. Every automation here defaults to draft until proven otherwise.

Advisory vs. autonomous

Advisory automations report and recommend; autonomous ones act without asking. Start everything advisory. Earn autonomy slowly, and only where a mistake is genuinely cheap to undo.

Chain trigger / message bus

One automation directly waking another when it finishes, rather than each one polling on its own schedule and waiting for its next slot. A later-stage optimisation, not a day-one requirement.

Self-review loop

A scheduled pass that reads back through recent automated work and proposes new standing instructions or fixes for a human to approve, so lessons get captured once instead of re-learned.

This document describes a real, working system as of August 2026. It will go stale, tools change, prices change, and this business's own setup keeps evolving. Treat it as a snapshot of an approach that worked, not a permanent spec.

Want help doing this in your business?

This playbook is the real build log from putting AI to work across a commercial trade business, not a theory. If you're a trade or services business owner and want a hand mapping this onto your own operation, get in touch.

Kieron Henare
Automation Integrator · kieron@ccsi.co.nz