Operations Handbook

Run the work you sell — intake, projects, tasks, work orders, service-desk tickets and service contracts — then track time, bill it and post the costs, without re-entry.

Version 1.0 · ixlcore.com

Reference

The Operations module is where the work your business delivers actually gets done. It takes a request in at one end, turns it into a project, a task, a work order or a support ticket, tracks the effort against it, and turns the delivered work into money owed at the other end. Because it sits on the same foundation as CRM and Accounting, the invoice you raise here is a real CRM invoice, the client is the same customer record you use everywhere, and the costs land in the ledger without a second keying.

This guide is a reference for what the module does and how the pieces fit together. It describes IXL CORE **version 1.0**.

Overview

Operations is a broad module. At a glance it covers:

- **Command centre** — a single operational read across projects, work, tickets and requests.
- **Intake** — a request queue that triages incoming work and converts it into the right record.
- **Projects** — the client work you plan, staff, deliver and bill, with phases, milestones, roles, risks, issues and status reports.
- **Tasks** — the units of work, with assignees, priorities, dependencies and a status flow.
- **Work orders** — field and site jobs with checklists, materials, scheduling and evidence.
- **Service desk** — support tickets with SLAs, queues, teams, macros and a public portal.
- **Service contracts** — retainers and coverage that sit behind ticket billing.
- **Billing & rate cards** — billable lines, rate resolution and CRM invoice requests.
- **Attendance** — field punches, devices, passkeys and CSV import feeding timesheets.
- **Governance, knowledge & directory** — risk/issue/decision registers, a knowledge base and a shared directory.

Everything is scoped to your company, entity and branch, and every action is governed by permissions (see Access & permissions).

Command centre

The **command centre** gives an operational summary in one place — the shape of live projects, open work, tickets in flight and requests waiting to be triaged. It can also produce a written **digest** so a manager can see, in plain language, what needs attention across the module without opening each area in turn.

Intake

Work usually starts as a **request**. You define **request types** for the kinds of work you take in, and each request carries a subject, description, priority and requester. A request follows a clear lifecycle — it is submitted, **triaged**, then **accepted** or **rejected**, **assigned** and finally **converted**. On submission the module runs a **duplicate check** so the same issue is not opened twice.

The important step is **convert**: an accepted request becomes the right kind of record — a **project**, a **task**, a **work order** or a **support ticket** — carrying its detail across so nothing is re-typed. Requests can route through the platform's shared **approval** engine where the work needs a sign-off before it proceeds.

Projects

A **project** is a piece of client work you plan, staff, deliver and bill. Each project carries a name, a client (a CRM account), a **billing type** — fixed price, time-and-materials or milestone-based — a currency, a **labour rate** reference and a project manager. It also tracks **health**, **percent complete** and start, target and actual-end dates. Projects move through a controlled **status lifecycle** and the transitions are enforced, so the state of every job stays clear.

A project is more than a header. It breaks down into:

- **Phases** — the stages the work moves through.
- **Milestones** — the billable deliverables, each with a due date, a **billing amount** and an **acceptance** step that can be routed for approval; an accepted milestone is what becomes eligible to bill.
- **Roles** — the team on the project, added with a role and removed without losing history.
- **Risks** and **issues** — a per-project register, each entry raised, updated and rolled into the project's **health**.
- **Status reports** — periodic write-ups of where the project stands.

A project also exposes **timesheets** (the effort logged against it) and a **health** view that reads budget and delivery signals together, so the project manager can see at a glance whether the job is on track.

Tasks

A **task** is a unit of work. Each has a title, an **owner** (it can belong to a project, a ticket, a work order or stand alone), an assignee, a **priority**, a due date and a source. Tasks move through a status flow —

open, in-progress and complete — and support **dependencies**, so one task can be marked as waiting on another. Tasks are created directly, spun off from tickets and work orders as fulfilment or follow-up steps, or generated from governance actions and meeting notes.

Work orders

A **work order** is a field or site job — the kind of work that happens away from a desk. Each carries a title, a category, a **billing type**, a priority, a site and location, a requester and customer, **required skills**, and a scheduled start and end. Work orders support:

- a **checklist** of steps to complete on site,
- a list of **materials** used on the job,
- **evidence** capture (for proof of work),
- **follow-up tasks**, and
- a **completion** step that can be routed for approval before the job is signed off.

Like everything else, work orders move through an enforced status flow and can be scheduled against your team's capacity.

Service desk

The service desk handles **support tickets** end to end. A ticket carries a reference, a subject and description, a **priority**, **impact** and **urgency**, a requester and customer, and it is routed to a **queue**, a **team** and an assignee. Beyond the basics it supports **replies** to the requester, **macros** (canned responses and actions), **evidence**, **fulfilment tasks** and a link to a **service contract** for coverage.

Tickets are governed by **SLA policies**. For each priority you set first-response and resolution targets, and the module calculates the **response-due** and **resolution-due** times, tracks first response and resolution, can **pause** the clock (for example while waiting on the customer), and flags a **breach** with escalation when a target is missed. A **ticket report** gives you the shape of the support workload.

The service desk is configured through its **admin** area: **queues**, **teams** and their members, **services**, **SLA policies**, **macros** and **assignment rules** that route tickets automatically. A **public ticket portal**, secured by a rotating portal key and per-ticket tokens, lets customers submit and follow up on tickets without a login.

Service contracts

A **service contract** is the coverage that sits behind support — a retainer or an entitlement for a customer. It records the covered services, a **coverage model**, **included minutes** or **included incidents** and what has been used against them, a retainer amount and currency, an SLA policy, and an effective period. Contracts are **activated** and **cancelled** through their own lifecycle, and where an SLA is missed a **service credit** can be recorded against the contract. **Service reviews** summarise how a customer's service performed over a period, with an optional AI-written narrative.

Billing & rate cards

Billing runs on **billable lines**. Effort, materials and expenses accrue against a project, work order or ticket as priced lines, each drawing its rate from a resolved source. A line can be edited, tagged or excluded before it is billed. When the work is ready to invoice, an **invoice request** is raised — routed through approval where required — and the result is a draft invoice in **CRM** for the customer, which posts to Accounting through the normal invoice flow. Nothing is billed twice: a line records the invoice it went onto.

Rates come from **rate cards**. A rate card is a versioned, publishable set of rates with a **default** and per-scope **assignments**, and the module **resolves** the right rate for a given piece of work from those assignments. Rate cards are stewarded as shared master data, so pricing stays consistent across the module rather than being re-entered per job.

Attendance

Attendance captures where and when field work happened, feeding the hours back into timesheets. Staff can **punch** from the field with location captured, register **biometric passkeys** for secure punching, and enrol physical **attendance devices** with rotating keys for site check-in. Batches can be brought in by **CSV import** with a template, preview and commit, and an **ingestion** endpoint accepts events from registered devices. Consent is captured and can be revoked, and confirmed punches flow through to HR timesheet entries.

Governance, knowledge & directory

Beyond delivery, Operations carries the supporting registers:

- **Governance** — organisation-wide **risk**, **issue** and **decision** registers plus **actions**, with a dashboard and approval routing for decisions.
- **Knowledge** — a knowledge base of **articles** with versions, a submit-for-publish approval, archiving, linking, ticket suggestions and **gap** detection.
- **Directory** — a shared read of staff, employees, contacts, accounts, CRM contracts, products and documents that other Operations screens draw on.

Scheduling

A **calendar** view schedules work across the team, with a **capacity** read and a **conflict check** so a person is not double-booked before an assignment is confirmed.

Access & permissions {#access-and-permissions}

Every Operations action is governed by a capability, and view and manage are separate — projects, tasks, tickets, work orders, intake, billing, service contracts, rate cards, attendance, governance, knowledge, scheduling and settings each have their own view and manage permissions. Capabilities are grouped into roles and roles are assigned to users, so each person sees and does only what their

role allows. This is enforced on every request, not just hidden in the interface, and sensitive figures such as cost and margin sit behind their own permission.

How Operations connects

Operations is where the work happens, but it does not stand alone. On the shared platform foundation:

- **CRM** receives every invoice request as a real invoice draft — by line, milestone, hour or material — so billing the work and getting paid use one connected record.
- **Accounting** receives the invoices you raise, so revenue lands in the ledger automatically.
- **HR** receives confirmed attendance punches as timesheet entries, so field hours become recorded time.
- **The customer record** is the same one used across the platform, so the client on a project, the customer on a ticket and the account on an invoice are one and the same.
- **The approval engine** governs milestone acceptance, work-order completion, decisions, article publishing and invoice requests through one shared workflow.

That connection is the point: you take the work in, deliver it, bill it once, and the money and the ledger stay in step.

How-to guides

Create and run a project

Set up a project, build out its phases and milestones, and assign the delivery team.

This guide covers creating an Operations project, structuring it into phases and milestones, and putting a team on it.

Before you start

- You need the operations projects-manage permission to create or edit projects (view-only users can browse).
- A project only needs a **name** to be created — everything else is optional.
- To name a project manager or add team members, those people must be members of your organisation.

Steps

1. Go to **Operations !’ Projects** and start a new project.
2. Complete the project fields:

[Screenshot: the new-project form]

3. Add **phases** to band the work. Each phase takes a **Name** (required), optional **Description**, a **Sort order**, and optional **Start** and **Target** dates. Phases are ordered by their sort value.
4. Add **milestones**. Each takes a **Name** (required), optional **Description**, **Status** (Pending, In progress, Missed), a **Due date**, a **Sort order**, a **Client visible** toggle, and may be tied to one of the project's phases.

[Screenshot: milestones listed under a phase]

5. Build the **team** from **Roles**. Pick a person and give them a role — **Manager**, **Member**, **Reviewer** or **Stakeholder**.
6. When a milestone is delivered, choose **Request acceptance** and name an approver (who must be someone other than yourself). This opens an approval and moves the milestone to **Awaiting acceptance**; it becomes **Accepted** only once that approval is granted.

Result

The project holds its phases, milestones and team, and its health is recomputed as milestones change. Milestone acceptance runs through the platform approval flow rather than by editing status directly.

Related

- Create and manage tasks
- Log and convert a work request
- Operations reference

Log and convert a work request

Define request types, capture an intake request, triage it, and convert it into a task, project, work order or ticket.

This guide covers setting up request types and running an intake request through triage and conversion.

Before you start

- You need the operations intake permission to submit and manage requests.
- Every request names a **request type**, so set your types up first.
- A request needs a requester: either a **staff member** in your organisation or a **CRM contact/ account** reference.

Steps

Set up request types

1. Go to **Operations !' Intake** and open **Request types**.
2. Complete the type fields:

[Screenshot: the request-type editor]

Log and progress a request

3.
4. Start a new request and complete:
   Submit the request (or save it as a draft first).
5. **Triage** and then **Accept** or **Reject** the request. Only accepted or triaged requests can be converted.

[Screenshot: triaging a request]

6. Choose **Convert** and pick the target: **Task**, **Project**, **Work order** or **Ticket**. Optionally set the assignee. The request is marked **Converted** and linked to the new record.

Result

The request moves through draft, submitted, triaged, accepted and converted states, producing a task, project, work order or ticket that carries the original context forward.

Related

- Raise and complete a work order
- Handle service-desk tickets with SLAs
- Operations reference

Create and manage tasks

Raise a shared Operations task, assign it, and track it through its lifecycle — on its own or against a project, work order or ticket.

This guide covers the shared Operations task: a single work primitive that can stand alone or belong to another Operations record.

Before you start

- You need the operations tasks permission to create and manage tasks.
- A task only needs a **title** to be created.
- To assign a task, the assignee must be a member of your organisation.

Steps

1. Go to **Operations !' Tasks** and start a new task.
2. Complete the task fields:

[Screenshot: the new-task form]

3. Optionally give the task an **owner** — the record it belongs to (for example a project, work order or ticket). The owner is set when the task is created and cannot be changed afterwards; a task with no owner stands alone.

[Screenshot: choosing a task owner]

4. Update the task's status and assignee as it progresses.

Result

The task is tracked through its lifecycle. Tasks created from a work order or ticket as a follow-up or fulfilment action are the same shared task, so they appear alongside the record that owns them.

Related

- Create and run a project
- Raise and complete a work order
- Operations reference

Raise and complete a work order

Create a work order, work its checklist, record materials and evidence, then request completion through approval.

This guide covers raising a work order, running it on site, and closing it out through the completion approval.

Before you start

- You need the operations work-orders permission to create and manage work orders.
- A work order only needs a **title** to be created.
- To assign one, the assignee must be a member of your organisation.

Steps

1. Go to **Operations !' Work orders** and start a new work order.
2. Complete the work-order fields:

[Screenshot: the new-work-order form]

3. Move the work order through its statuses — **Draft**, **Scheduled**, **In progress**, **On hold** — as work proceeds.
4. Build the **checklist**. Each step has a **Label** and can be marked **Required**; steps are ordered by their sort value. Tick steps off as you go and attach notes or document references to a step.

[Screenshot: the work-order checklist]

5. Add **materials** — references to Supply Chain products with a quantity. This records what was used; no stock is moved.
6. Attach **evidence** as document references, and add **follow-up actions**, which are created as shared Operations tasks owned by this work order.
7. Choose **Request completion** and name an approver. Every **required** checklist step must be done first. This opens an approval and moves the work order to **Awaiting completion**; it becomes **Completed** only when the approval is granted (or returns to In progress if rejected).

Result

The work order carries its checklist, materials and evidence, and is completed through the platform approval flow rather than by editing status directly.

Related

- Log and convert a work request
- Capture field attendance
- Operations reference

Handle service-desk tickets with SLAs

Raise, route and resolve service-desk tickets, reply to customers, apply macros, and share a secure portal link.

This guide covers working service-desk tickets: routing them to queues and teams, replying, applying macros and tracking the SLA clock.

Before you start

- You need the operations tickets permission to create and manage tickets.
- A ticket needs a requester: a **staff member** or a **CRM contact/account** reference.
- Queues, teams, services and SLA policies must already exist in your organisation to route to them.

Steps

1. Go to **Operations !’ Service desk** and start a new ticket.
2. Complete the ticket fields:

[Screenshot: the new-ticket form]

3. **Assign** the ticket to an agent, and move it through its statuses — **New**, **Open**, **Pending**, **On hold**, **Resolved**, **Closed**.
4. **Reply** on the ticket. A reply is either **Public** (customer-visible) or an **Internal** note, and may carry document references.
5. **Apply a macro** to insert a canned reply and optionally set the status or assignee in one step.
6. Attach **evidence** as document references, and add **fulfilment actions**, created as shared Operations tasks owned by the ticket.

[Screenshot: replying to a ticket with the SLA clock showing]

7. Share a **portal link** by minting a portal token, optionally protected by a passcode and an expiry. The link is shown once so you can copy it; revoke it at any time.

Result

The ticket is routed, worked and resolved with its SLA state tracked (on track, at risk or breached). The ticket report summarises volumes by status, priority and queue, and SLA attainment.

Related

- Set up a service contract
- Log and convert a work request
- Operations reference

Set up a service contract

Review a provisioned service contract, set its coverage and allowance, and activate it to govern support tickets.

This guide covers reviewing and activating an Operations service contract and understanding its coverage and credits.

Before you start

- You need the operations service-contracts permission to manage contracts.
- Service contracts are provisioned automatically from a signed CRM support agreement, so you review and activate rather than create from scratch.
- Any SLA policy or covered services you attach must belong to your organisation.

Steps

1. Go to **Operations !’ Service contracts**Contracts pending review are listed first.
2. Open a **pending-review** contract and set its terms:

[Screenshot: the service-contract review form]

3. Choose **Activate** to move the contract from **Pending review** to **Active**. An included-hours contract must carry a positive included-minute allowance. Activation is blocked if it would conflict with another active contract for the same customer.
4. On an active contract, review the **drawdown** (used against included minutes and incidents, with remaining minutes shown), the **SLA credits** applied, and the linked CRM account and contract.
5. Choose **Cancel** to stop a contract governing new tickets.

[Screenshot: an active contract showing drawdown and credits]

Result

An activated contract governs its customer's tickets, drawing down its allowance and recording any SLA credits. Its status runs through pending review, active, expired or cancelled.

Related

- Handle service-desk tickets with SLAs
- Set up rate cards and bill work
-

Operations reference

Set up rate cards and bill work

Author a rate card of priced lines, publish it, set a default, and assign customer-specific overrides.

This guide covers building an Operations rate card, publishing it as an immutable version, and overriding it per customer.

Before you start

- You need the operations rate-card permission to author, publish and assign rate cards.
- A rate card is versioned: editing a published card cuts a new draft version, leaving the published one untouched.
- Money on a rate line is entered as a decimal amount, never a raw float.

Steps

1. Go to **Operations !’ Rate cards** and start a new card.
2.
3. Complete the card header:
   Add at least one **rate line**. Each line prices an activity, role or service:

[Screenshot: the rate-card editor with priced lines]

4. Choose **Publish** to freeze the open draft version. A published version is immutable; the card's header points at it.
5. Choose **Make default** on a published card so it applies where no override exists.
6. Assign a **customer override** by picking a **CRM account** and the **rate card** to use for it, with an optional note. There is at most one assignment per account; re-assigning replaces it.

[Screenshot: assigning a rate card to a customer]

Result

Published rate cards resolve the applicable rate for an activity, role or service, honouring a customer override first, then the default. Editing a published card cuts a fresh draft to publish next.

Related

- Set up a service contract
- Raise and complete a work order
- Operations reference

Capture field attendance

Clock staff in and out by GPS or passkey, register ingestion devices, manage consent, and import timesheets by CSV.

This guide covers the ways Operations captures time — field GPS punches, phone-biometric passkeys, keyed devices and CSV upload — all feeding the existing HR timesheet system.

Before you start

- Recording your own attendance needs the operations attendance view permission; registering devices, enrolling employees, managing consent and importing timesheets need the manage permission.
- To clock in, your login must be linked to a staff record.
- No biometric template, fingerprint, face or image is ever stored — only device metadata, the device-to-employee mapping, consent records and punch events.

Steps

Clock in and out

1. Go to **Operations !' Attendance**Use **Field clock** to record a punch: choose a **Direction** (**In** or **Out**), which may carry your **location** (latitude, longitude and accuracy) and an optional link to a work order, ticket or task. A field punch only ever records your own attendance.
2. To clock in by phone biometric, register a **passkey** (a WebAuthn credential you can label), then use it to punch in and out. A backdated or forward-dated time outside the accepted window is rejected.

[Screenshot: the field clock with direction and location]

Register devices and consent (admin)

3. **Register a device** — give it a **Name**, a **Kind** (biometric, NFC, QR, mobile or other), a **Location label** and a **Timezone**. The ingestion key and signing secret are shown once; copy them into the device. Rotate or deactivate the device later as needed.
4. **Enrol** an employee on a device by mapping them to the device's external user id, and **record consent** for an employee (with a purpose and version) before biometric capture. Consent can be revoked.

[Screenshot: device registration showing the one-time key]

Import timesheets by CSV

5. Download the **CSV template**, fill it in, then **Preview** the upload to check it, and **Commit** to record the punches.

Result

Punches from every source roll up into the existing HR timesheet system rather than a separate time silo. Unmapped device punches are queued for you to reconcile, and the overview lists devices and recent punches.

Related

- Raise and complete a work order
- Create and manage tasks
- Operations reference