
Province Improvement Prompt Pack

Copy-ready implementation prompts based on the useful Province-wide ideas identified in The Overseer feature overview, adapted to your existing Order - Province - Unit platform.

Run in the stated sequence. Every prompt preserves tenant isolation, server-side authorisation, multi-Order context, secure documents, audit history, public URLs and the shared dashboard.

Delivery order

1. Governance and scope
2. Health dashboard
3. Forms tracker
4. Role and feature matrix
5. Secure roll-up reporting
6. Contacts and templates
7. Communications, tasks and alerts
8. Calendar, search and cross-membership
9. Optional independent-Unit sharing

Use: paste one prompt into a new implementation task. Do not combine prompts until the prior prompt's migrations, authorisation tests and deployment checks pass.

1. Province data-sharing and access-transparency contract

CORE

Foundation for every Province-wide report, search, email or document feature.

You are working on my existing PHP 8.2/MySQL Lodge Management Platform. Inspect Province and Unit links, active_user_memberships, scoped_permission_grants, TenantContext, ProvinceContext, dashboard navigation, audit logs, exports, search, reports, documents and welfare APIs before changing anything.

Implement an explicit server-derived Province data-sharing contract for membership, contact details, events, attendance, candidates, finance aggregates, welfare aggregates, documents and group email. Require all applicable controls: the Unit relationship permits the category, the Province feature is enabled, and the actor has the scoped capability. Record sensitive/personal-data access with actor, scope, category, purpose, timestamp and source workspace. Provide Unit-facing transparency, revocation and fresh evaluation on every route/API/export/download. Do not infer access from membership, expose raw IDs, or let reports bypass Unit/Province scope. Deliver migrations, a capability matrix, tests and rollback.

Completion gate: confirm fresh server-side scope enforcement, audit evidence, migration/backfill safety and relevant automated tests before proceeding.

2. Action-led Province health dashboard

CORE

Build on existing Province operational reporting; do not create a duplicate dashboard.

You are working on my existing PHP 8.2/MySQL Lodge Management Platform. Inspect the Province dashboard, operational-dashboard API, reports, calendar, returns, scheduled-job health, finance, welfare, candidates and attendance before changing anything.

Add a capability-filtered Province health strip and a Units-needing-attention list. Include only authorised aggregates: no future meeting, overdue approved return/form, emergency-contact gap, unresolved candidate stage, unconfirmed attendance, overdue finance item and failed/stale job. Every tile must state its definition and as-of time and link only to authorised detail. Suppress sensitive small groups and distinguish unavailable data from poor performance. Sort Units by actionable risk. Do not expose welfare case detail, personal finance detail or cross-Province data. Add responsive/browser, API-scope and role tests.

Completion gate: confirm fresh server-side scope enforcement, audit evidence, migration/backfill safety and relevant automated tests before proceeding.

3. Province forms and returns tracker

CORE

Addresses the largest operational workflow gap.

You are working on my existing PHP 8.2/MySQL Lodge Management Platform. Inspect annual returns, reporting schedules, notifications, Province roles, audit logs and document storage before changing anything.

Implement a governed Province forms tracker for recurring returns and approved Almoner/administrative forms. A form definition needs an owner, Order/Unit scope, due-date rule, required fields, submission status, review state, critical flag, reminder schedule and retention rule. Show submitted, outstanding, overdue and critical status by Unit without exposing restricted contents. Preserve versioned submissions, allow authorised correction/review, send capability-safe reminders and audit every submission/review. Do not use silent deletion or spreadsheet-only tracking. Add migration, scheduler, access-control and overdue-reminder tests.

Completion gate: confirm fresh server-side scope enforcement, audit evidence, migration/backfill safety and relevant automated tests before proceeding.

4. Province roles, defaults and feature switches

CORE

Use the existing capability model; do not create parallel permissions.

You are working on my existing PHP 8.2/MySQL Lodge Management Platform. Inspect role assignments, scoped permission grants, capability mapping, Province navigation, APIs, exports, documents, finance, welfare and scheduled reports before changing anything.

Complete an explicit Province role matrix for Administrator, Secretary, Treasurer, Almoner, Director of Ceremonies, Charity Steward and Viewer. Define separately who may discover, view, export, edit, approve and schedule each feature. Add Province feature switches for finance, welfare, contacts, documents, reports and communications. A disabled feature must disappear from navigation and be denied server-side. Preserve custom grants, expiry, revocation and existing roles. Do not give Province access merely because a user has a Unit office. Add route/API/export/download tests and a migration plan.

Completion gate: confirm fresh server-side scope enforcement, audit evidence, migration/backfill safety and relevant automated tests before proceeding.

5. Secure Province roll-up reports

NEXT

Prioritise definitions and reconciliation before charts.

You are working on my existing PHP 8.2/MySQL Lodge Management Platform. Inspect the report catalogue, finance ledger, candidates, attendance, calendar, welfare, person identity, exports and scheduled reports before changing anything.

Add governed Province roll-up reports for membership/cross-membership, attendance, candidate pipeline, finance aggregates, charity/alms aggregates, welfare health, forms status and ritual coverage where source data exists. Define every measure, source query, filter, aggregation, suppression threshold and as-of timestamp. Keep welfare and finance detail separately permissioned; never export private notes or Unit transaction detail by default. Use verified person identity only after review, not fuzzy automatic merges. Preserve Unit financial periods and existing report encryption, audit and scheduling. Add query-volume, cross-Province, role, export and reconciliation tests.

Completion gate: confirm fresh server-side scope enforcement, audit evidence, migration/backfill safety and relevant automated tests before proceeding.

6. Province contacts and versioned template distribution

NEXT

Consistency without taking control from Units.

You are working on my existing PHP 8.2/MySQL Lodge Management Platform. Inspect people/contacts, officer roles, meeting documents, templates, email templates, summons workflow, document versions, Unit links and audit logs before changing anything.

Implement controlled Province publishing. Maintain Province team contacts with office, contact method, published state and receives-summons flag. Publish authorised records into selected Units with source provenance, local override protection and revocation. Let Province publish approved summons layouts, email templates or meeting planners as editable Unit copies. Track master version, local version, acceptance, replacement and rollback; never silently overwrite a Unit's local changes. Every write into a Unit must be capability checked and audited as a Province-originated action. Add scope, replacement, revocation and local-customisation tests.

Completion gate: confirm fresh server-side scope enforcement, audit evidence, migration/backfill safety and relevant automated tests before proceeding.

7. Province communications, tasks and alerts

NEXT

Keep communications, tasks and alerts separately governed.

You are working on my existing PHP 8.2/MySQL Lodge Management Platform. Inspect mail queue, delivery preferences, consent/suppression, action centre, scheduled jobs, notification templates, Province roles and audit logs before changing anything.

Add three governed tools: a Province group emailer with recipient preview, selected Units/roles, consent and suppression enforcement, individual delivery, rate limits, attachment permissions and full audit; a Province task/helpdesk workflow with Unit/member links, owner, due date, status, notes, escalation and safe Unit visibility; and alert rules/daily digests for approved operational events. Do not use unrestricted bulk email, expose welfare details in alerts or auto-email people without policy. Add role, queue, retry, unsubscribe and audit tests.

Completion gate: confirm fresh server-side scope enforcement, audit evidence, migration/backfill safety and relevant automated tests before proceeding.

8. Province calendar, scoped search and cross-membership

NEXT

Build only after the data-sharing/access-log contract is live.

You are working on my existing PHP 8.2/MySQL Lodge Management Platform. Inspect Province calendar, calendar feeds, member workspace, search endpoints, document search, people identity, event routes and capability checks before changing anything.

Extend the Province experience with an authorised combined calendar, private iCal subscriptions, scoped search and a cross-membership view. Calendar feeds must contain events only, never attendee/member data. Search must label results by Unit and Order, apply the data-sharing contract before matching, and not reveal results through counts or autocomplete. Cross-membership must use verified person links and show a review queue for ambiguity rather than automatic merges. Retain active Order context and opaque workspace navigation. Add browser, cache, feed-token, leakage and cross-Province tests.

Completion gate: confirm fresh server-side scope enforcement, audit evidence, migration/backfill safety and relevant automated tests before proceeding.

9. Optional: independently administered Unit sharing

OPTIONAL

A strategic product decision, not required for centrally managed Units.

You are working on my existing PHP 8.2/MySQL Lodge Management Platform. Inspect tenant boundaries, Province/Unit links, invitations, audit logs, consent, subscription configuration and data-governance records before changing anything.

Design and implement opt-in sharing for independently administered Units only if this is an approved product direction. A Unit owner chooses categories to share, generates a one-time approval token with optional expiry, sees who accessed shared data, revokes access instantly and receives expiry notices. Province access is read-only by default. Any Province-originated write, such as a task, document or template, needs separate approval, capability checking and a Unit audit entry. Do not treat sharing as a replacement for tenant isolation or assume a Province link grants consent. Add token, expiry, revocation, audit, disclosure and cross-tenant tests.

Completion gate: confirm fresh server-side scope enforcement, audit evidence, migration/backfill safety and relevant automated tests before proceeding.

Suggested delivery gates

Phase	Prompts	Gate
Stabilise	1, 4	Scope matrix, feature switches, transparency and direct-route tests pass.
Operational core	2, 3, 5	Definitions reconcile with Unit records; reporting and reminders are auditable.
Province tools	6, 7	Publishing, communications and tasks have consent, review and rollback.
Discovery	8	No feed/search leakage; active Order and Province context remain intact.
Expansion	9	Stakeholders approve independent-Unit sharing and controller policy.

Global guardrails

Preserve tenant isolation, fresh server-side authorisation, existing routes/public URLs, Order - Province - Unit relationships, multi-Order context, secure documents, audit history, responsive behaviour and existing Unit workflows. Hidden navigation is never authorisation.