PC Form & PCN Workflow Audit, Data Integrity Review, and Unified Flow Plan

Date: 22 August 2026
Scope: Codebase at `d:\Projects\performance_management_system` (Laravel 10 / PHP 8.2, Blade UI, MySQL/`pc` connection)
Constraint: No production code was changed. This document is analysis only.

Evidence labels used throughout:

- **Confirmed** — verified in schema, models, services, routes, or tests
- **Strongly inferred** — consistent across multiple call sites but not documented as policy
- **Uncertain** — insufficient evidence in the repository

---

1. Executive Summary

The application does **not** have a single PC Form / PCN lifecycle. It has **parallel products** that share naming (`state`, `status`, `is_revised`, `is_modified`, draft tables, meetings) but are **sibling entities under a scheme (`projects.project_id`)**, not parent/child.

Critical facts (**Confirmed**):

1. **There are no incrementing `revision_number` or `modification_number` columns.** Revision and modification are **boolean flags** plus **new database rows** (`parent_id` chain). Display "revision position" is computed in listing code, not stored.
2. **Drafts are implemented at least three ways** for PC Forms (in-place `pc_forms.state = draft`, standalone `pc_forms_drafts` with `pc_form_id` null, and linked `pc_forms_drafts` rows that **accumulate** on save).
3. **Two approval machines coexist:** actor-based **forward/decline** (`workflow_stage`, `current_actor_id`, `pc_form_forwards` / `pcn_forwards`) for PC-I/PC-II and PCN, versus **legacy counters** (`check_count` / `approve_count`) still used for later PC kinds and leftover rows.
4. **PCN is not generated from a PC Form.** Meetings attach **either** a PC Form **or** a PCN via `mws_meeting_subject_links`.
5. **Import is PCFMS-only** (`pc:migrate-pcfms` → `PcfmsPcMigrationService`). It writes directly to `pc_forms`, maps source IDs in `pcfms_import_maps`, and **does not** use `PcFormService::create()`. Imported `parent_id` is often set to **the row's own id**. PCN import was **not found**.

The recommended target is: **one scheme-scoped document lifecycle, one draft store, one versioning model (immutable submitted rows + flags only as derived labels), one workflow engine, import that exits into that model.**

---

2. Scope and Methodology

Inspected:

- Repository layout: Laravel `app/`, Blade `resources/views/`, `routes/web.php`, `database/migrations/`, `tests/`
- Models: `App\Models\PC\PcForm`, `PcFormDraft`, `ProjectConceptNote`, `ProjectConceptNoteDraft`, `PcfmsImportMap`, `MwsMeetingSubjectLink`
- Services: `PcFormService` (~14k lines), `ProjectConceptNoteService`, `PcfmsPcMigrationService`, workflow/stage/recipient/approval-count services
- Controllers: `PcFormsController`, `ProjectConceptNoteController`
- Seeders: `PcFormControlAccessSeeder`, `ProjectConceptNoteControlAccessSeeder`
- Tests under `tests/Feature/PcForm/`, `tests/Unit/*PcForm*`, `tests/Unit/*Pcn*`
- Global searches for revision/modification/draft/import/meeting/permission identifiers

Frontend is **server-rendered Blade + jQuery/Select2**, not a separate SPA. There is no dedicated React/Vue PC Form app.

---

3. Repository Findings

Layer	Finding	Confidence
Backend	Laravel 10, `PcFormsController` + `PcFormService`	Confirmed
Frontend	`resources/views/pc-forms/`, `resources/views/project-concept-notes/`	Confirmed
Database	`pc_forms` + many section tables; mirrored `pc_forms_drafts` + `pc_form_draft_*`; `project_concept_notes` + section tables (split later)	Confirmed
PC kinds	`pc_i`, `pc_ii`, `pc_iii_a`, `pc_iii_b`, `pc_iv`, `pc_v` on `pc_forms.pc_form_kind`	Confirmed
PCN	Independent module; same scheme; similar flags/workflow	Confirmed
Meetings	`mws_meetings` + `mws_meeting_subject_links` (`pc_form_id` XOR `project_concept_note_id`)	Confirmed
Import	Artisan `pc:migrate-pcfms`; map table `pcfms_import_maps`	Confirmed
Permissions	`control_access` + `can('pc-forms', action)` / `can('project-concept-notes', action)`	Confirmed

`PcFormService.php` is the gravitational center of PC Form behavior. PCN is smaller and cleaner but **copied the same flag semantics**.

---

4. Current State — Core Concepts

4.1 What each word actually means

Concept	What it is in this codebase	Separate table?	Status?	Number?	Confidence
PC Form	One row in `pc_forms` plus section child tables	Yes (`pc_forms`)	`state` + `status` (active/inactive)	UID `pc_uid`	Confirmed
PCN	One row in `project_concept_notes` plus section tables	Yes	Same pattern	`pcn_uid`	Confirmed
Draft (PC new)	`pc_forms` row with `state = draft`	Same table	Yes	No	Confirmed (`2026_08_20_214202_add_draft_to_pc_forms_state_` `PcFormService::create($isDraft=true)`)
Draft (standalone)	`pc_forms_drafts` with `pc_form_id` null	Yes	`state=draft`	No	Confirmed
Draft (linked)	`pc_forms_drafts` with `pc_form_id` set; **new row on every "save as draft"** when in-place save is not used	Yes	Yes	No	Confirmed (comment in `PcFormService::update`)
PCN draft	`project_concept_note_drafts` standalone or linked	Yes	Yes	No	Confirmed
Revision	New `pc_forms` / `project_concept_notes` row with `parent_id` = root id, often `is_revised=true`	New row	Plus workspace flags	**No stored number**	Confirmed
Modification	Boolean `is_modified` (and PCN `modified_pcn_source_id` workspace)	Flag / extra row	Not a status	**No stored number**	Confirmed
Version	PCFMS `VersionNo` used only to **sort import groups**; not persisted on `pc_forms`	No	No	Import-only	Confirmed
Submission	`status=1`, `submitted_at` set, workflow started	Same row or new row	`state` / `workflow_stage`	No	Confirmed
Approval / rejection	Reviews (`pc_form_reviews` / `pcn_reviews`) **and/or** forward actions; `state` synced from stage	Review tables	Yes	Counts on legacy path	Confirmed
Meeting	MWS meeting + subject link	Yes	Meeting lifecycle separate	N/A	Confirmed
Imported form	Normal `pc_forms` row + `pcfms_import_maps`	Map table	Mapped from PCFMS status	Source VersionNo discarded	Confirmed
System-generated	User-created via wizard; also PC-II–V dedicated create routes	Same	Same	Same	Confirmed
Historical record	Prior chain rows with `status=0`; PCFMS files via `PcfmsPcformsHistoricalFilesService`	Same / files	Inactive	Computed position	Confirmed
Snapshot	Not a first-class snapshot table; copies of section rows	Copied children	N/A	N/A	Confirmed
Audit	`LoggingService` / activity events; reviews; forwards	Activity + review + forward	N/A	N/A	Confirmed

4.2 Current vs original vs "current version"

- **Original / root:** `parent_id` **null** in live create path (`PcForm::isRoot()`). **Import often sets** `parent_id = id`**.** (**Confirmed** — `PcfmsPcMigrationService::importPc1Groups` / `importPc2FromPc1`)
- **Current submitted:** `status = 1` (tinyint/boolean "active"). Older chain members set `status = 0` on revision spawn / import of non-latest.
- **Current workspace:** unpublished shells: `post_approval_revision_workspace=1` and `status=0` (PC); PCN also `modified_pcn_source_id` + `status=0`.
- **Multiple rows can represent one logical scheme document.** Listing code specially includes workspaces and filters "current" (**Confirmed** — `PcFormService` listing helpers).

---

5. Current PC Form Workflow

5.1 Entry points (**Confirmed**)

Action	HTTP	Controller	Service
Index / list	`GET /pc-forms`	`PcFormsController@index`	`PcFormService` listing + standalone drafts
Create PC-I	`GET/POST /pc-forms`	`create` / `store`	`PcFormService::create`
Create PC-II–V	`GET pc-forms/create-pc-ii` … `create-pc-v`	dedicated create	same `create` with kind
Edit / update / submit	`GET/PUT /pc-forms/{id}`	`edit` / `update`	`PcFormService::update`
Autosave draft	`POST pc-forms/draft-autosave` (+ `/{id}`)	`autosaveDraft*`	service draft persist
Delete form	`DELETE /pc-forms/{id}`	`destroy`	`PcFormService::delete`
Delete standalone draft	`DELETE pc-forms/drafts/{draft}`	`destroyDraft`	`deleteStandaloneDraft`
Start revised PC-I	`GET pc-forms/begin-approved-revision`	`beginApprovedRevision`	`spawnPostApprovalRevisionWorkspace`
Revision gate	`GET pc-forms/revision-availability`	`revisionAvailability`	service
Forward / decline	`POST pc-forms/{id}/forward|decline`		`PcFormWorkflowService`
Approve / reject (legacy + meeting)	`POST .../approve|reject`		`PcFormService::approve` / reject
Chairman advise/defer	`POST .../chairman-*`		`storeChairman*`
Submit modified to CDS	`POST .../submit-modified-to-cds`	`submitModifiedToCds`	service
Import	Artisan `pc:migrate-pcfms`	`MigratePcfmsPcCommand`	`PcfmsPcMigrationService::migrate`

5.2 Create / draft / submit (happy path)

```
Wizard POST (isDraft)
  → PcFormService::create(true)
  → INSERT pc_forms state=draft, persist sections on MAIN tables via persistDraftPcFormWorkspace

Wizard POST (submit)
  → PcFormService::create(false)
  → INSERT pc_forms, section tables, submitted_at, workflow defaults
  → completeForwardWorkflowAfterSubmit() if PC-I/II
```

Confirmed: first-time draft now lives on **`pc_forms`**, not `pc_forms_drafts`.

5.3 Saving / editing drafts

Situation	Mechanism	Confidence
`state=draft` or revision workspace or rejected/modified editable	**In-place** update of `pc_forms` + section tables (`pcFormAllowsInPlaceDraftWorkspaceSave`)	Confirmed
Existing submitted form, “save as draft”, not in-place eligible	**Always INSERT new** `pc_forms_drafts` row (explicit comment)	Confirmed
Never submitted, listing “standalone”	`pc_forms_drafts.pc_form_id` null; `createMainFormFromStandaloneDraft`	Confirmed
Autosave	Same persist path as draft workspace	Confirmed

Saving a draft **does not increment a revision counter** (none exists). It **does** write activity logs on in-place save.

5.4 Submit / resubmit

`PcFormService::update(..., $isDraft=false)`:

1. If `shouldCreateRevisionOnSubmit()` → **new row** via `createPcFormRevisionFromSubmit`, source typically inactivated.
2. Else if latest linked draft `state=draft` → `copyDraftToMain`, draft → `requested`.
3. Else patch the same `pc_forms` row from the request.

`shouldCreateRevisionOnSubmit` (**Confirmed**):

- Post-approval revision workspace → always new row
- Else only if `state=rejected` **and** latest rejection in cycle is **approver-type**, **and** not same-row defer (checker/department/PnD/chairman defer)

5.5 Approve / reject / forward

Forward path (PC-I / PC-II when `usesForwardWorkflow()`): actor-based. Stages in `PcFormWorkflowStageResolver`: `ad_review`, `ad_approve`, `pnd_development_section`, `pnd_internal_circulation`, `ddwp_department_handoff`, `meeting_linked`, `approved`, `declined`. `syncLegacyState()` writes old `state` enum.

`usesForwardWorkflow()` (**Confirmed**, `PcForm.php`): PC-I/II only if `status` truthy and `state` not draft/rejected/approved. Inactive workspaces therefore **do not** use forward until submitted.

Counter path: `PcFormApprovalCountService` increments `check_count` / `approve_count` and advances `state` (`requested` → `checking` → `department_approval_pending` → `under approval` → `approved`). Still used for PC-III+ unless `uses_forward_workflow` column is set.

Reviews live in `pc_form_reviews` with expanding action enums (defer, chairman, AA, concurrence, etc.).

5.6 Meeting association

- Link table: `mws_meeting_subject_links.pc_form_id`
- `PcForm::hasLinkedMeeting()`
- Stage can become `meeting_linked`
- Defer/hold flags: `defer_repeat_*` / `defer_block_repeat_*` on `pc_forms` change **whether a new working paper** may be generated (`allowsNewWorkingPaperGeneration`) — **meeting type is a workflow discriminator** (**Confirmed**)

5.7 Workflow map vs requested list

#	Flow	Status
1	New PC Form	Implemented — `create`
2–4	Draft create/save/edit	Implemented — **multiple mechanisms**
5	Submit	Implemented — `create`/`update`
6	Revise submitted	Implemented — new row / workspace
7	Modify	Implemented as **flag + same-row or new-row resubmit**, not a numbered series
8	Approve/reject	Implemented — dual machines
9–11	PCN create/modify/revise	Implemented in PCN module (see §6)
12–13	Meeting create/associate / type behavior	Implemented in MWS + flags
14	Generate from another source	**PC-II–V from scheme/PC-I context**; not "generate PCN from PC Form"
15–20	Import lifecycle	**Ingest only** via PCFMS; after that they are `pc_forms` with map rows. No separate imported-edit API
21	System-generated	Same create path
22	Clone/copy/duplicate	**Not a user "duplicate" action.** Internal `copyMainFormToRevision` / `copyDraftToMain` / rejected clone
23	Reopen submitted	Via reject/defer edit, revision workspace, or modified flags
24	Cancelled/withdrawn	**Not implemented / not found** as first-class states
25	Archived	Working papers mention "not archived"; PC Form **archive state not found**
26	Historical	Inactive chain rows + PCFMS file browser
27	Other	AI analyze; cover/AA/advice letters; post-forum; actor reconcile commands; workflow count sync commands; **EAED mobile API** for PC-I location drafts (`/api/v1/eaed`, `EaedDraftService`)

6. Current PCN Workflow

6.1 Entry points (**Confirmed**, `routes/web.php`)

Resource `project-concept-notes` plus: approve, reject, forward, decline, timeline, workflow-actions, revision-availability, **begin-approved-revision**, **begin-modified-revision**, preview, scheme-eligibility.

6.2 Create / draft / submit

`ProjectConceptNoteService::create`:

- Draft → **standalone** `project_concept_note_drafts` (`project_concept_note_id` null)
- Submit → `project_concept_notes` + section persist + `finalizeSubmitAndForward`

Linked draft: `createLinkedDraftFromMain` — **new draft row each time** (same accumulation pattern).

Difference vs PC Form: first PCN draft is **not** an in-place `project_concept_notes` row (PC Form first draft **is** in-place on `pc_forms`). **Confirmed.**

6.3 Revision vs modification

Action	Method	Persistence
Revised PCN (after approved)	`beginApprovedRevision`	New row, `status=0`, `post_approval_revision_workspace=1`, `is_revised=1`, copy sections
Modified PCN	`beginModifiedRevision` → `beginModifiedPcnRevision`	New row, `status=0`, `modified_pcn_source_id`, `is_modified=1`

Action	Method	Persistence
Submit workspace	`shouldCreateRevisionOnSubmit` includes modified workspace	New **active** row

Eligibility: `revisionAvailability()` — no in-progress PCN + approved exists; modified baseline separate.
Modified PCN additionally requires an **active PCN plus a completed linked meeting** (`pcnQualifiesForModifiedRevision` / `pcnHasCompletedMeeting`) — **Confirmed**.

6.4 Workflow stages

`PcnWorkflowStageResolver`: similar AD/PnD/meeting path; **terminal** `cleared_cdwp` instead of generic approved-only. Deprecated stages (`working_paper`, `pre_pdwp`, ...) remain as constants (**Confirmed**).

6.5 PCN import

Not implemented / not found in inspected code (no PCN analog of `PcfmsPcMigrationService`).

---

7. Draft Lifecycle (deep)

Current State

PC Form

- Authoritative **wizard drafts:** `pc_forms` (`state=draft`).
- **Legacy / linked / standalone:** `pc_forms_drafts` + full parallel section schema (migration `2026_02_10_174449`).
- Drafts are **mutable**.
- **Multiple linked drafts per form are allowed and intended** ("do not update existing draft").
- Submit copies **latest** `state=draft` via `latest('id')` — older linked drafts become stale leftovers.
- Standalone delete: `destroyDraft` / `deleteStandaloneDraft`. Linked drafts: cascade if `pc_form_id` FK `nullOnDelete` when parent deleted.
- Abandoned drafts: **no expiry job found**.
- Imported drafts: import writes **main** `pc_forms`, not draft tables (**Confirmed**).
- **EAED (mobile):** `routes/api.php /api/v1/eaed/drafts*` → `EaedDraftService` creates/updates **standalone** `pc_forms_drafts` (location/photos). Fourth write path into the draft table (**Confirmed**). Web promote remains `createMainFormFromStandaloneDraft`.

PCN

- Parallel draft tables; first save is standalone draft.
- Modified/revision workspaces are **main table shells**, not draft table.

Problems

- Two physical draft models on PC Forms; listing merges them (`is_standalone_draft` stub).
- Latest-id submit can **ignore** a newer in-place main-row edit if a stale linked draft exists (**Strongly inferred** from `latest('id')` precedence in `update`).
- No optimistic lock on drafts.
- Parallel section tables double schema cost and drift (PC-III/IV/V columns added on both sides over many migrations).

Is the draft model logically sound?

No. It is historically layered: shadow tables → in-place workspace → leftover "always insert new draft". Sound rule would be: **one mutable working copy per (user, logical document) until submit.**

---

8. Revision Lifecycle (deep)

What causes a revision? (**Confirmed**)

Trigger	PC Form	PCN
First submit on scheme that already had approved PC-I	`is_revised` on **new** create (`shouldMarkNewPcAsRevisedSchemeFirstSubmit`)	N/A (blocked if in progress)
Post-approval "Revised"	Workspace row then submit clones active row, `is_revised`	Same
Approver-stage reject (not defer)	New row; `is_revised` unless defer-repeat flags force "modified" semantics	Same pattern
Import later VersionNo / later PC1 in group	`is_revised=true`, `parent_id=root`	Not imported
Display	`revision_chain_position` computed by `id` order	No equivalent found

Does not increment on: draft save, meeting change, metadata-only edit, import VersionNo persistence (VersionNo discarded).

Storage

- `parent_id` → root id (live) or **self** (import)
- `is_revised` boolean
- `post_approval_revision_workspace`, `revision_source_approved_id` (PC Form)

- Full **row copy** of section tables (`copyMainFormToRevision`)
- Source row `status=0` when superseded (**import** and revision submit)

Scope: per scheme + kind (PC-I chain). Not global. Not a numeric sequence stored.

Problems

- `is_revised` is overloaded (post-approval Revised PC-I **and** some resubmit clones **and** import).
- Computed position can disagree with business "Revision 2" if ids are not chronological or import `parent_id=self` pollutes grouping (`attachRevisionChainPositions` treats `parent_id ?? id` as root).
- No unique constraint on `(parent_id, …)` "one active revision".

---

9. Modification Lifecycle (deep)

What qualifies? (**Confirmed**)

- **Not** a field-level change detector. No "which columns count" engine.
- `is_modified` set on:
  - Successful resubmit after certain rejections (`resolveIsModifiedOnSuccessfulResubmit`) — elevated, **never cleared**
  - New revision row from approver reject (`is_modified = !fromPostApprovalWorkspace`)
  - PCN modified workspace from day one
- **Same-row** resubmit after checker/department/PnD/chairman **defer** — routing uses `shouldSameRowResubmitAfterDefer`; `is_modified` may stay false for checker/dept defer
- Meeting hold flags **do not** set `is_modified` but **change WP generation** when `is_modified` is already true

Numbering

There is no modification counter. Duplication/skip/reset of numbers **cannot occur as numbers**; inconsistency is **boolean vs row history** (modified without a new row, or new row without clear user-facing "Mod N").

Draft save does **not** set `is_modified`. Import does not set a modification series (only `is_revised` for later PC1s).

---

10. Version / History Storage

Mechanism	Role	Authoritative?
Extra `pc_forms` / PCN rows	Prior submitted editions	Yes for content
`pc_form_reviews` / `pcn_reviews`	Decisions	Yes for actions
`pc_form_forwards` / `pcn_forwards`	Actor trail	Yes for forward workflow
Activity / `LoggingService`	Audit text	Partial
`pcfms_import_maps`	Source identity	Yes for import identity
Files `pcfms_pcforms`	Legacy binaries	Yes for files, not structured data
JSON on forms	Content fields, not version snapshots	Content

No event-sourced revision log. No JSON snapshot of full form per version.

---

11. Meeting Workflow

Meetings are a **separate product** (Meeting Workspace). For PC/PCN they:

- Attach subjects (`MwsMeetingSubjectLink`)
- Gate MoM / PC decisions (see workspace MoM rules; PC approve/defer can require MoM signature completeness — **Confirmed** in project rules and meeting tests)
- Drive `workflow_stage = meeting_linked`
- Store **per-meeting-type** repeat/block flags on the **form/PCN row**

Meeting type as hidden workflow discriminator — Confirmed: `allowsNewWorkingPaperGeneration` switches on `pre-pdwp` / `pdwp` / `ddwp` columns. Forum decision tests (`PdwpForumDecisionFlowTest`) encode type-specific PC actions.

Recommendation: keep meeting type as a **permission + policy dimension for MWS**, but **stop storing meeting-policy flags as ad-hoc booleans on the document** if a unified model is adopted (move to meeting/hold records).

---

12. Permission Model

Current State (**Confirmed**)

- Stored: `control_access` (module/form/route/action), role grants, field rows `control_access_fields` for **"PC 1"** only.
- Checked: `can($routeSlug, $action)` (`app/helpers.php`).
- PC Form kinds **share** `pc-forms` actions. **Not** split by `pc_form_kind`.

- PCN: `project-concept-notes` list/view/add/edit + workflow-oriented actions; seeder **removes** a long list of legacy PCN and leaked PC-I actions (`ProjectConceptNoteControlAccessSeeder`).
- Meeting types: **separate** MWS control-access (MoM generate, etc.), not the same catalog as PC Form.
- Forward actor: **not** only `can_approve` — `current_actor_id` + recipient resolver (hardcoded role/stage rules in `PcFormWorkflowRecipientResolver` / `PcnWorkflowRecipientResolver`).
- Department scope: `PcFormApprovalCountService` overlap helpers reused by PCN.
- Frontend: Blade `can()` hides buttons; backend also checks — **not frontend-only**, but UI and service rules can diverge (actor vs `can_check`).

Should permissions split by PC Form type and meeting type?

Yes, but only two extra dimensions, not a matrix of every verb × every kind × every forum.

Recommended smallest model:

```
Role
  → grants
      → document_class: pcn | pc_i | pc_ii | pc_iii | pc_iv | pc_v
      → actions: list, view, create, edit_own_draft, submit, delete_draft,
                 revise_approved, modify_resubmit, check, department_approve, pnd_act, approve
  → meeting_class: pre_pdwp | pdwp | ddwp | general
      → actions already owned by MWS (create meeting, WP, MoM, circulate)
```

Do not add separate permissions for draft-save vs autosave, or import (import is ops/CLI).

Why split kinds: PC-III–V are singleton-per-scheme and skip PC-I sections (`isPerSchemeSingletonKind`, `isShortPcWizardKind`). One "add PC 1" grant already unlocks all create-* routes if the controller only checks `can('pc-forms','add')` (**Confirmed** on `beginApprovedRevision`).

Why not split every meeting action onto the PC Form grant: meetings already have MWS permissions; duplicating them on PC Form created the PCN legacy action sprawl the seeder is deleting.

Drawback of over-granularity: this app already has field-level PC-I view/edit (~40 fields × 2). Combining that with kind × meeting × action will make roles unmaintainable.

---

13. Imported PC Form Workflow

Pipeline today (**Confirmed**)

```
pcfms.sql
  → PcfmsSqlTableReader
  → PcfmsSchemeResolver (SC_UID / ADP → projects.project_id)
  → group PC1 by scheme, sort VersionNo then PC1ID
  → upsertPcFormFromPc1 / upsertPcForm
  → pcfms_import_maps (source_table, source_id, pc_form_kind) UNIQUE
  → section imports (authorities, objectives, …)
  → PC2/PC3/PC4/PC5 sibling forms, often state=approved
```

Re-import: mapped rows **updated**; workflow counts **preserved** (`PRESERVED_WORKFLOW_COLUMNS`, command comment).

`mapWorkflowState`: PCFMS `PC1StatusID` 4/5/7 → `rejected`; `1` → `requested`; **default** `approved`.

Imported → normal flow issues

Issue	Root cause	Consequence	Fix	Data cleanup
`parent_id = self`	Import sets root to own id	`isRoot()` false; chain math treats self as parent	Normalize `parent_id` null for roots	UPDATE where parent_id=id
Inactive historical rows `status=0`	Latest-only active	Correct-ish, but mixed with live revision rules	Keep; document	None
Default state `approved`	Loose status map	Forms skip forward workflow (`usesForwardWorkflow` false when approved)	Map unknown to `requested` or explicit imported_approved	Review unmapped status IDs
No `workflow_stage` / actor	Bypass `PcFormService`	Cannot forward until reconciled	`ReconcilePcFormForwardWorkflowCommand` / actor commands	Run reconcile
VersionNo dropped	Not mapped to a column	Cannot show PCFMS version	Optional `source_version` on map table	Backfill from SQL if kept
PC-III/IV/V forced approved + counts=1	Import payload	Appear finished; may not match reality	Import as historical approved **explicitly** flagged	`origin=pcfms`
No PCN import	None written	PCN always "system"	Out of scope or separate job	N/A

Issue	Root cause	Consequence	Fix	Data cleanup
Historical files	Separate storage service	OK if paths exist	Keep	N/A
Edit/submit after import	Same controllers	**Works if state/status allow**; approved locked (`isPostAaAuthLocked` / approved)	Treat imported approved as terminal unless user starts Revised	None
Duplicate PCN	N/A	Import does not create PCN	—	—
ID collision	New autoincrement ids; source in map	Safe	Keep map unique	—

End state goal (imported becomes normal): Safe for content, if roots are normalized and imported approved rows are explicitly **historical-complete** (same as a live approved PC-I). **Not safe** to drop `pcfms_import_maps` (re-import idempotency).

---

14. PC Form → PCN Relationship

Confirmed:

- **Not** parent/child. No `pc_forms.pcn_id`. PCN has `project_id`, not `pc_form_id` on the note (meeting links have both columns).
- One scheme can have **many** PC Form rows (kinds + revision chain) and **many** PCN rows (chain + workspaces).
- One PCN does **not** reference a PC Form revision.
- Changing a PC Form after PCN exists **does not** update the PCN.
- PCN is a **live row** (plus copied sections on revise), not a snapshot of a PC Form.
- Imported PC Form **can** later have a user-created PCN on the same scheme (no code block found that forbids it).
- Meeting change does not rewrite PCN↔PC Form FKs (there are none).
- **Meeting decisions share one pair of routes** for both document types: `POST meeting-workspace/meetings/{id}/linked-pc-form/approve` and `.../defer` (`MeetingWorkspaceMeetingController` → `MwsPcFormMeetingDecisionService` / `ProjectConceptNoteService::approveFromMeetingWorkspace`). MoM signature gate applies per subject (`MwsMomSchemeSubject`: exclusive `pc_form_id` **or** `project_concept_note_id`). **Confirmed.**

Referential risks: two in-flight documents on one scheme; meeting link hydrates missing PCN by **latest non-terminal PCN for project_id** (`hydrateMissingProjectConceptNotes`) — can attach the **wrong** PCN (**Confirmed**).

---

15. Functionality-to-Workflow Matrix

Functionality	Current flow	Tables	Draft	Revision	Modification	PCN	Meeting	Import
Create PC-I	`PcFormService::create`	`pc_forms`+sections	In-place or none	Flag if scheme already approved	No	Independent	Later	No
Create PC-II–V	Same + kind routes	Same	Same	Singleton kind rules	No	Independent	Later	PCFMS siblings
Edit draft	In-place or new draft row	Main or drafts	Multiple	No	No	Parallel	—	—
Submit	`update`/`create` + forward complete	Main	Convert latest linked	Maybe new row	Maybe flag	`finalizeSubmitAndForward`	—	—
Approve/reject	Forward **or** counters + reviews	reviews, forwards	—	—	Defer vs reject	Same split	Forum actions	Bypass
Revised	Workspace + submit clone	New `pc_forms`	In-place on shell	`is_revised`	Cleared on shell	Same	WP rules	Import `is_revis`
Modified	Flag + same/new row	Same	In-place if rejected	Sometimes new row	Boolean	`modified_pcn_source_id`	Hold flags	No
Import	CLI upsert	`pc_forms`, map	Unused	Chain by VersionNo	No	None	None	Authoritat path
Generate WP	Working paper services	`working_papers`	—	New row allows WP	Hold if modified	PCN WP type	Type-specific	—
Clone	Internal copy only	Sections	—	Copy	—	Copy sections	—	—
Historical files	File service	disk	—	—	—	—	—	PCFMS

Functionality	Current flow	Tables	Draft	Revision	Modification	PCN	Meeting	Import
Reopen	Reject/workspace	Same	Yes	Yes	Yes	Yes	Defer flags	Edit if not locked
Cancel/archive	Not found	—	—	—	—	—	—	—

16. Duplicate / Competing Flows

Duplicate	Where	Why it exists	Who uses it	Keep?
In-place draft vs `pc_forms_drafts`	`PcFormService`	Schema first, UX later	Listing + update	**Merge to in-place + optional standalone only for never-submitted**
Linked "always new draft" vs in-place	`update($isDraft)`	Comment-era behavior	Older edit path	**Deprecate insert-always**
Forward vs counter workflow	I/II vs III+ and legacy	Migration mid-flight	`usesForwardWorkflow`, reconcile commands	**One engine**; counters derived
`state` enum vs `workflow_stage`	Model + resolver sync	Compatibility	All listings	**Stage authoritative**; state generated
PC `is_revised` vs PCN same flags	Copied design	Feature parity	Both UIs	Keep concept, **one shared policy class**
Approve POST vs forward POST	Controller	Old checkers + new actors	Both live	**Forward is canonical** for I/II/PCN; keep approve for meeting/forum
PCFMS upsert vs `PcFormService::create`	Command vs HTTP	Bulk legacy	Ops	Keep ingest; **normalize then stop special-casing**
Meeting hydrate PCN by project	`MwsMeetingSubjectLink`	Missing FK history	MWS	**Require explicit PCN id**

17. State Machine

PC Form `state` values (**Confirmed**)

`draft`, `requested`, `checking`, `department_approval_pending`, `under approval`, `approved`, `rejected`
(`draft` added later; original enum lacked it.)

Parallel `workflow_stage` (PC-I/II)

`ad_draft` (const) / `ad_review` → `ad_approve` → `pnd_development_section` → `pnd_internal_circulation` → (`ddwp_department_handoff`) → `meeting_linked` → `approved` | `declined`

`PcFormWorkflowStageResolver::syncLegacyState` maps these onto `state`.

Who moves

- Submitter: draft → requested / first actor
- Current actor: forward/decline
- Meeting/forum: approve, defer, chairman actions (reviews)
- Import: jump to approved/rejected/requested **without transitions**

Illegal / messy transitions

- Import → `approved` without forwards (**Confirmed**)
- `status=0` workspace with `state=requested` looks "in flight" to naive queries
- Rejected + `is_modified` + same-row resubmit resets counts but keeps flags
- Frontend may show Draft for workspaces while `state` is `requested` (`PcFormService` listing comments ~1580)

Canonical target machine (recommended)

```
draft → submitted → in_review → (returned_to_creator) → submitted
                          → meeting
                          → approved
                          → declined
approved → (optional) revision_draft → submitted … (new edition row)
```

Store **one** `lifecycle_status`. Derive listing labels in one resolver.

18. Database / Table Audit

Table	Purpose	Authoritative?	Used by	Duplicate?	Problems	Recommendation
`pc_forms`	Document header + many kind columns	Yes	HTTP, import, MWS	Kind-specific JSON/text on same row	God table	Keep; stop adding policy flags
`pc_form_*` sections	Body	Yes	Service persist	Mirrored in drafts	Schema clone	Keep; generate persist
`pc_forms_drafts` + `pc_form_draft_*`	Shadow body	Partial	Listing, old save	**Full duplicate of sections**	Drift	**Deprecate** except standalone pre-create
`pc_form_reviews`	Decisions	Yes	Approve/forum	Overlaps forwards	Action enum growth	Keep
`pc_form_forwards`	Actor trail	Yes	Forward WF	—	—	Keep
`pc_form_department_approvals`	Dept sign-off	Yes	Counter/dept	—	—	Keep until engine merge
`project_concept_notes`	PCN header	Yes	PCN HTTP	Was wide; split	Flags copy PC	Keep
`pcn_*` section tables	Body	Yes	Persistence	Draft twins	—	Keep
`project_concept_note_drafts` + draft sections	Drafts	Partial	Create/update	Duplicate	Same as PC	Same deprecate path
`pcn_reviews` / `pcn_forwards`	WF	Yes	PCN	—	—	Keep
`pcn_department_approvals`	Created in `2026_07_27_120000`	**Uncertain / unused**	No Eloquent model found	Likely leftover of PC Form dept table	Dead table risk	**Confirm row counts; drop or wire**
`pcfms_import_maps`	Source map	Yes	Re-import	—	—	**Keep forever**
`mws_meeting_subject_links`	Subject	Yes	MWS	Hydration fallback	Nullable FKs	Require one subject FK
`working_papers`	WP	Yes	MWS/PC	`pc_form_id` / `project_concept_note_id`	—	Keep
`control_access*`	Perms	Yes	`can()`	Field-level explosion	PC 1 only fields	Keep actions; trim fields
Activity/logs	Audit	Partial	LoggingService	Not full snapshot	—	Keep

19. Column Audit (important headers)

`pc_forms` — keep

`id`, `pc_uid`, `project_id`, `pc_form_kind`, content columns, `created_by`/`updated_by`, `submitted_at`, timestamps.

Dual-purpose / problem columns

Column	Issue	Depends on	Replace with	Migration
`status` (0/1)	Means **active edition**, not workflow	Listing, import, `usesForwardWorkflow`	`is_current` or `superseded_at`	Backfill from status
`state`	Legacy + synced from stage	Everywhere	Generated from `workflow_stage`	Keep column temporarily
`workflow_stage`	Real machine for I/II	Forward	Canonical	Backfill III+
`parent_id`	Null vs self	Root/chain	Null = root; CHECK	Fix import
`is_revised`	Overloaded boolean	UI, WP	Derived: `edition_kind=revision`	Optional new enum
`is_modified`	Sticky boolean	WP hold, UI	`edition_kind=modification` or review reason	Don't clear history
`post_approval_revision_workspace`	Workspace vs edition	Listing, submit	`row_role=workspace`	
`revision_source_approved_id`	Good FK	Spawn	Keep	Index already
`check_count` / `approve_count` / `required_*`	Second machine	Legacy + import	Derived from reviews	Freeze writes

Column	Issue	Depends on	Replace with	Migration
`defer_repeat_*` / `defer_block_*`	Meeting policy on document	WP generate	Meeting/hold table	Move later
`uses_forward_workflow`	Kind opt-in	III+	Remove when all forward	
`workpaper` longText	Old notes	Some UI	WP table	Deprecate

Not present: `revision_number`, `modification_number`, `version`, `import_id` on `pc_forms` (import id is map table).

`project_concept_notes`

Same flag set plus `modified_pcn_source_id`, `workflow_stage`, `current_allowed_actions`, `cleared_for_cdwp_at`. Same recommendations.

---

20. Data Integrity Risks

Referential

- `parent_id` FK exists (`nullOnDelete`) — **Confirmed**
- Import self-parent **violates the live root convention** (FK still valid)
- Draft `pc_form_id` nullOnDelete — orphans become standalone
- Meeting links can lack both FKs and hydrate by project — **wrong subject**
- Section tables generally `cascadeOnDelete` — good

Version

- No unique "one current per (project, kind)" except business rules in PHP (`isPerSchemeSingletonKind`, eligibility)
- Concurrent `begin-approved-revision` can create two workspaces (delete-then-insert in transaction per user, **not** global unique)
- `revision_chain_position` not stored → listing-only

Modification

- Flag without history row
- History rows without incrementing mod id (by design)

State

- Dual `state` + `stage`
- Import approved stuck outside forward
- `status=0` + `state=requested` workspace

Concurrency (**Confirmed / Strongly inferred**)

- Create/update use `DB::transaction` — good
- **No** `updated_at` version token / lockForUpdate on submit
- Two users: last write wins; two submits can both pass eligibility if no unique constraint
- Import in one transaction per migrate — good
- Linked draft `latest('id')` race: two drafts, submit picks max id

---

21. Concurrency / Transaction Recommendations

Atomic (already partly transactional — **extend locks**):

Operation	Need
Submit / create revision	Transaction + `lockForUpdate` on current row + unique `(project_id, kind, is_current)`
Spawn workspace	Unique partial index: one open workspace per `(project_id, kind, created_by)` or per scheme
Create PCN	Same uniqueness for in-flight PCN per scheme
Import upsert	Keep unique `(source_table, source_id, kind)` — **already exists**
Increment counters	Eliminate or serialize via review insert only
Meeting link	Unique subject per meeting

Prefer **optimistic** `lock_version` on `pc_forms` / PCN for editor conflict. Use **pessimistic** only on submit/revision spawn.

---

22. Coding Issues (severity)

Sev	Issue	Evidence
Critical	Dual workflow + dual draft can lose or mis-apply submit payload (`latest` draft vs in-place)	`PcFormService::update`

Sev	Issue	Evidence
Critical	Import `parent_id=self` + default `approved`	`PcfmsPcMigrationService`
High	`PcFormService` god-class (~14k lines); business rules in controller + service	File size, `PcFormsController`
High	Meeting subject hydration guesses PCN	`MwsMeetingSubjectLink::hydrateMissingProjectConceptNotes`
High	No optimistic concurrency	Absent
High	`FOREIGN_KEY_CHECKS=0` on draft management-structure insert	`PcFormService` ~8124
Medium	Magic strings for stages/states	Resolvers exist but many raw comparisons
Medium	Hardcoded meeting type columns	`allowsNewWorkingPaperGeneration`
Medium	Permissions: all kinds under `pc-forms add`	Controller
Medium	N+1 risk on listing with many relations	`show()` eager list is huge
Medium	Silent catch around revision workspace logging	`spawnPostApprovalRevisionWorkspace`
Low	Deprecated PCN stage constants still present	`PcnWorkflowStageResolver`
Low	`workpaper` column vs WorkingPaper model	Dual

23. Flow / Business Issues (not mere bugs)

1. **Revision and modification are flags + row copies, but the product language implies numbered editions.**
2. **Imported forms are the same table but not the same lifecycle** until reconciled.
3. **PCN and PC Form look like a pipeline; they are peers.** Users can run both on one scheme without a defined handoff.
4. **Meeting type changes document behavior** (WP, defer hold) without being a formal workflow type.
5. **Draft and submitted rules differ by kind and by which draft store is used.**
6. **PC-I/II vs PC-III–V** are different products sharing one controller/service.

24. Frontend vs Backend

- Blade `can()` hides create/revise; backend repeats `can()` and service exceptions.
- Revision/modified eligibility is **AJAX** (`revision-availability`) then redirect — backend is source of truth (**good**).
- Listing **relabels** workspace as Draft while DB `state` is `requested` — UI/backend status mismatch (**Confirmed** comments in `PcFormService`).
- Autosave vs full submit: backend section-gated persist (`draftWorkspaceShouldSave`) — rules live in PHP; JS must send the same section keys or data is skipped (**Strongly inferred**).
- Business rules in JS (PCN modified resume in `crud-scripts.blade.php`) **mirror** service messages — keep backend authoritative.

Move to domain: eligibility, revision spawn, submit row-split, `is_modified` elevation — already mostly in services; **stop adding new rules only in Blade.**

25. Recommended Unified Architecture

Target State (not current)

```
Scheme (project)
  └── Document (type: pcn | pc_i | …)
        ├── editions[]          immutable after submit
        ├── working_copy        0..1 mutable (draft / returned / workspace)
        ├── workflow            one engine (actor + stage)
        ├── reviews / forwards  append-only
        └── origin              user | pcfms (+ map)
Meetings reference edition_id (or working_copy only if explicitly allowed)
```

Principles:

- One lifecycle per document type
- One draft mechanism (working_copy)
- One edition mechanism (new row on publish of a new edition)
- `is_revised` / `is_modified` become **edition_reason**, not independent counters
- Import **normalizes then exits**
- Meeting type is MWS policy, not a second PC state machine

26. Recommended Revision / Modification Model

What is a revision?

A new submitted edition of an already approved (or import-historical approved) document, started as an explicit “Revised” action.
First-ever submit is edition 1, not a revision.

What is a modification?

A change to an in-flight or deferred document that remains the same edition (same public identity / same `pc_uid` row), typically after return/defer.
If policy requires a **new legal instrument** after approver reject, that is a **new edition** with reason `resubmit_after_approver_reject` — do not call it both revised and modified.

Deterministic increments (target)

There is **no business need for two integer counters** given current code. Recommend:

- `edition_no` integer, per `(project_id, document_type)`, assigned **only when a working_copy is published** (`submitted_at` set, `is_current=1`).
- `edition_reason`**:** `initial` | `revised_after_approval` | `resubmit_after_return` | `imported`.
- Draft save → **no** edition_no change.
- Meeting / attachment / metadata on a draft → **no** edition_no change.
- Import historical rows → assign edition_no from PCFMS `VersionNo` **or** 1..n by sort order, then freeze.

Same action ⇒ same edition behavior for imported and native after ingest.

Drop sticky `is_modified` for **new** writes; keep column as generated: `edition_reason = resubmit_after_return`.

---

27. Recommended Import Architecture

```
External SQL
  → staging (optional tables or temp)
  → validate scheme map
  → normalize (parent_id null, edition_no, origin=pcfms, lifecycle_status)
  → write canonical pc_forms + sections
  → write pcfms_import_maps
  → stop
Normal HTTP workflow only
```

- Staging: drop after success; **map table is permanent**
- Source IDs: **only** in `pcfms_import_maps`
- Duplicates: existing unique key
- Re-import: update content, **do not** reset `edition_no` or destroy reviews
- History: inactive editions `is_current=0`
- Errors: fail the scheme group, not half a chain (today: one big transaction — **keep**)
- Idempotency: source unique key (**already correct**)

---

28. Recommended Permission Architecture

See §12. Final:

- **Keep** route-level `can('pc-forms'|'project-concept-notes', action)`
- **Add** document_class on create/revise (or separate routes already exist — grant per route)
- **Keep** MWS meeting-type permissions where they are
- **Do not** duplicate MWS verbs onto PCN/PC Form
- **Shrink** field-level grants to section-level if roles cannot be administered
- Actor assignment remains **data** (`current_actor_id`), not a permission

---

29. Keep / Merge / Deprecate / Drop

Component	Keep	Merge	Deprecate	Drop	Reason
`pc_forms` + sections	✓				Authoritative
`pc_forms_drafts` linked path			✓	later	In-place working_copy
Standalone drafts	✓ short term	into working_copy without parent edition			UX for unsaved new
`is_revised` / `is_modified`	read	into `edition_reason`	writes		Flags overloaded
`state`	read	from stage			Dual machine
`workflow_stage` + forwards	✓	extend to III+			Canonical WF
Counter columns	read		✓ writes	later	Legacy
`PcFormService`		split by use-case			Maintainability
`PcfmsPcMigrationService`	✓ ingest	output via normalizer	special states		Needed

Component	Keep	Merge	Deprecate	Drop	Reason
`pcfms_import_maps`	✓				Idempotency
PCN services	✓	shared edition policy			Parallel OK
`hydrateMissingProjectConceptNotes` persist			✓		Wrong-link risk
Approve POST (I/II)	meeting/forum		actor approve		Forward canonical
Clone product feature				N/A	Not found
Event sourcing				✓ don't add	Overkill

30. Target Data Model (logical)

```
DocumentEdition          ← authoritative submitted (or imported historical)
  project_id
  document_type          pcn | pc_i | …
  edition_no
  edition_reason
  is_current
  lifecycle_status
  origin                 user | pcfms
  parent_edition_id      previous edition (linear), not self
  created_by, submitted_at

DocumentWorkingCopy      ← 0..1 mutable (can be same table with row_role=working)
  edition_id nullable    (null = first create)
  same sections

WorkflowAssignment
  current_actor_id, stage, allowed_actions

Review / Forward         ← append-only

ImportMap                ← source_table + source_id + document_type → edition_id

MeetingSubjectLink       ← meeting_id + edition_id (type implied)
```

- **Current:** `is_current=1` edition, or working_copy if open
- **History:** other editions
- **Immutable:** submitted editions' section rows (new edition = copy-on-write)
- **Mutable:** working_copy only
- PCN and PC Form **share this shape**; they stay separate types, not one table, unless you later unify (not required)

31. Migration Strategy

Phase 1 — Audit (measure)

- Count `parent_id = id`, `parent_id is null`, multiple `status=1` per (project, kind)
- Count linked drafts per `pc_form_id`
- Count rows with `workflow_stage` null vs counters-only
- PCFMS status ID histogram vs `mapWorkflowState`
- Open workspaces per scheme

Phase 2 — Introduce canonical columns

Add nullable `edition_no`, `edition_reason`, `origin`, `row_role`, `lock_version` **without** switching writes.

Phase 3 — Normalize existing data

- Roots: `parent_id = id` → `null`
- `edition_no` = `revision_chain_position` or import order
- `origin=pcfms` if map row exists
- `row_role=working` if workspace or `state=draft`

Phase 4 — Imported records

- Tag origin; set `lifecycle_status=approved` only when PCFMS says so
- Do not invent forward trails
- Allow `begin revised` using same spawn as native approved

Phase 5 — Application

- New submits only create working_copy + publish edition
- Stop inserting linked `pc_forms_drafts` on save
- Forward engine for all kinds that still use HTTP workflow

Phase 6 — Deprecate

- Stop writing counters except backfill
- Listing ignores draft table except leftovers

Phase 7 — Remove

- Drop unused draft section tables after zero rows
- Drop `uses_forward_workflow` if universal

Rollback: keep old columns; feature-flag publish path; import map unchanged.

---

32. Backward Compatibility

Must remain readable:

- All existing `pc_forms` / PCN ids (WP, meetings, letters, MoM)
- `pc_uid` / `pcn_uid`
- Review/forward history
- PCFMS file links
- Inactive chain rows

Do **not** keep insert-always drafts or self-`parent_id` as **behavior** — keep **rows**.

---

33. Testing Strategy

Existing coverage is **workflow/forward/scope** heavy, **not** edition/import integrity (`tests/Unit/PcFormRevisionForwardSubmitTest`, `PcnModifiedRevisionTest`, forward feature tests).

Unit

- Edition assignment on publish
- Draft save does not change edition_no
- `parent_id` normalization
- `mapWorkflowState` exhaustive PCFMS ids
- Permission: PC-II create without PC-I add (once split)

Integration

- Create → draft → submit → forward → meeting → approve
- Approved → revised workspace → submit → new edition current
- Defer → same-row resubmit → edition_no unchanged
- Import → map → HTTP edit rules
- PCN modified workspace → publish
- Meeting link cannot hydrate wrong PCN

Integrity

- Unique current edition
- Unique import source
- Concurrent submit (two processes)

Critical regressions (never break)

1. PC-I first submit assigns actor
2. Post-AA-auth lock on approved row
3. Revised workspace not locked by parent AA
4. Defer hold WP generation per meeting type
5. PCFMS re-import does not wipe counts (until counters retired, then do not wipe reviews)
6. MoM signature gate for PC decisions (existing product rule)
7. Standalone draft delete does not delete a submitted form
8. Scheme eligibility on **create**, not same-scheme resubmit

---

34. Risk Assessment

Risk	Level	Mitigation
Listing/WP/MoM keyed by `pc_forms.id`	High if you renumber	**Never change ids**
Silent wrong PCN on meeting	High	Stop hydrate-persist
Import default approved	High	Status map + origin
God-class rewrite	Medium	Extract, don't rewrite in place
Dual-write period	Medium	Feature flag
Over-fine permissions	Medium	Cap dimensions

---

35. Final Recommended Flow (plain language)

1. An officer **creates** a PCN or a PC form (I–V) for a scheme (or scheme-less PCN).
2. They **save a working copy** as often as they want. That does not create a new official edition.
3. They **submit**. That working copy becomes the **current edition** and enters **one** review chain (forward to the next responsible officer).
4. Reviewers **forward, comment, or return**. Return lets the creator edit the **same edition**.
5. If a forum **defers**, the same edition is corrected and sent again. Meeting "hold" is a meeting policy, not a new form type.
6. When the edition is **approved** (or PCN cleared for CDWP), it is locked.
7. A later **Revised** action copies the approved edition into a new working copy. Submit creates **edition N+1** and makes it current. Old editions stay visible and unused for new meetings unless explicitly chosen.
8. **Imported PCFMS** records are loaded as editions with origin "imported". After load they follow the same rules: approved stays locked; Revised starts edition N+1 the same way.

PCN and PC-I are **two document types on the same scheme**, not "PCN produces PC Form automatically." A meeting is attached to a **specific edition**, not "the scheme in general."

```
Create → Working copy → Submit (edition N)
       → Review / Return (same edition)
       → Meeting (optional)
       → Approved (lock)
       → Revised working copy → Submit (edition N+1)
```

---

36. Executive Decision Checklist

Keep

- `pc_forms` / PCN / section tables as content stores
- Forward trail + reviews
- `pcfms_import_maps` and CLI ingest
- MWS meetings with **explicit** subject FKs
- `revision_source_approved_id` / `modified_pcn_source_id` as lineage FKs

Merge

- Draft stores → one working copy
- `state` + `workflow_stage` → one lifecycle
- PC vs PCN edition policy → shared rules
- Flag pair revised/modified → `edition_reason`

Deprecate

- Insert-always `pc_forms_drafts`
- Counter-driven workflow writes
- Meeting-policy booleans on the document
- Persist-on-hydrate meeting PCN

Drop (later)

- Unused draft section tables
- Self-`parent_id` convention
- Dead PCN stage constants (code only)

Introduce

- `edition_no` + `is_current` + unique constraint
- `origin`, `row_role`, `lock_version`
- Import normalizer (roots, stages)
- Kind-scoped create permission
- Domain services extracted from `PcFormService`

Highest-priority bugs

1. Submit may apply **wrong draft** (`latest id` vs in-place)
2. Import **self-parent** + **default approved**
3. Meeting **wrong PCN** hydration
4. Race on two revision workspaces / two current rows

Highest-priority flow problems

1. Two approval machines
2. Two draft machines
3. Revision/modification are not numbers but product language pretends they are
4. Import forever special unless normalized
5. PCN/PC Form assumed pipeline — they are peers

Highest-priority data problems

1. Multiple “current” interpretations (`status`, workspace, listing filters)
2. `parent_id` semantics split live vs import
3. Sticky `is_modified` without edition identity
4. No uniqueness for one in-flight document per scheme/kind

Recommended permission model

Role → document class (PCN, PC-I, PC-II, PC-III, PC-IV, PC-V) → small action set; meeting class stays in MWS. **Do not** split every verb by meeting type on the form itself.

Recommended revision model

`edition_no` **on publish only**; linear `parent_edition_id`; first submit = 1; Revised after approval = N+1.

Recommended modification model

Same edition, reason `returned`/`deferred`. No second integer. Stop using `is_modified` as a second version axis.

Recommended import model

Map → normalize → canonical editions → **same HTTP lifecycle**. Keep source keys. Do not keep a second workflow.

Recommended target architecture

```
Scheme
  ├─ PCN editions + 1 working copy + 1 workflow
  └─ PC-{kind} editions + 1 working copy + 1 workflow
        └─ Meetings / WP / MoM reference edition_id
Import ──normalize──► same editions
```

---

37. Confidence and gaps

Topic	Confidence	Gap
No revision/modification integers	Confirmed	—
Dual draft + dual workflow	Confirmed	—
PCN not child of PC Form	Confirmed	Product intent (should they be sequenced?) **Uncertain** — not in code
PCN import	Confirmed absent	Future requirement unknown
Exact PCFMS `PC1StatusID` meaning	Strongly inferred from match()	Official PCFMS dictionary not in repo
Whether production data already has `parent_id=id`	Strongly inferred (code always writes it on import)	Needs Phase 1 SQL
Cancel/withdraw/archive	Confirmed not found as states	Soft delete / `status=0` only
API vs web	Confirmed web-centric for full CRUD	Sanctum **EAED** API writes standalone PC-I drafts only (`/api/v1/eaed`). No full PC/PCN REST CRUD. Product FR: `docs/PC-Form-Management-Functional-Requirements-Workflow.md` (may lag live forward-chain / MoM rules)

End of audit.