Assessments Module – Developer Overview

Architecture • Data Model • Flows • APIs

For internal developer use

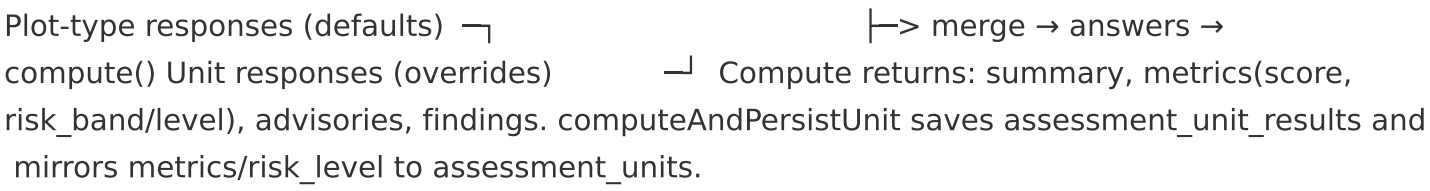
Big picture

- An Assessment = one “job” a user starts (e.g., Fire FRA). - A Unit = one individual property being assessed (flat/house). It lives in assessment_units. - A Plot Type = a template for many similar units (e.g., “Type A” for flats 1–30). Stored in assessment_plot_types. - Responses live at two levels: • Plot-type responses: defaults for all units in that plot (assessment_plot_type_responses). • Unit responses: per-property overrides (assessment_unit_responses). - Compute & persist writes per-unit metrics/advisories into assessment_unit_results and mirrors latest metrics/risk_level on assessment_units. - Reports (PDF + snapshot meta) go in assessment_reports (kind='plot' or 'unit'). Preview pages should read these.

Core tables (what they’re for)

1) assessment_types - Catalog of product types (e.g., fire). Keys: id, slug, current_version. 2) assessments - Parent container for a job. Keys: id, uuid, assessment_type_id, schema_version, status, meta, metrics, risk_level. 3) assessment_sites - Optional site (address/postcode) for an assessment. Used to seed unit addresses in plots. 4) assessment_plot_types - Plot template (“Type A / Block A”). Keys: id, assessment_id, name/reference, meta. 5) assessment_plot_type_responses - Default answers for a plot-type per step_slug; data JSON per row. 6) assessment_units ← INDIVIDUAL PROPERTY LIVES HERE - One per flat/house. Keys: id, uuid, assessment_id, plot_type_id?, status, meta{label,address,postcode,plot_identifier}, metrics, risk_level. 7) assessment_unit_responses - Per-unit answers by step_slug; data JSON per row (FK: assessment_unit_id — no assessment_id here). 8) assessment_unit_results - Canonical persisted compute output for a unit (metrics, advisories, answers, qa). 9) assessment_reports - Stored PDFs + snapshot meta for plot or unit: path, url, meta, kind('plot'|'unit'), plot_type_id?, unit_id?.

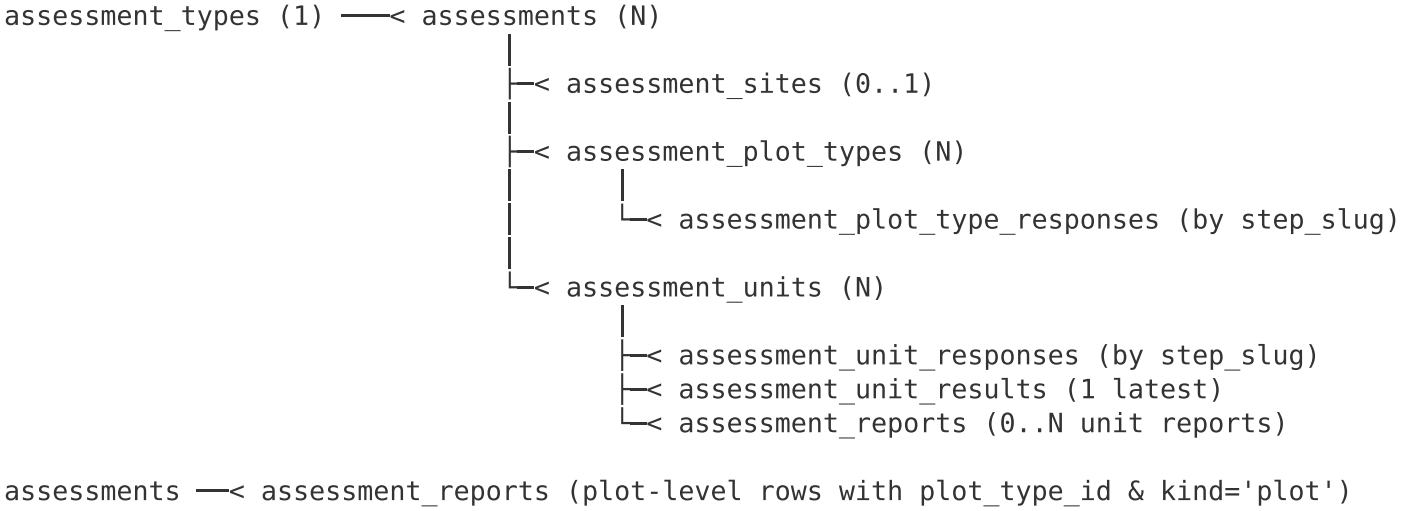
How answers flow into compute



Bands & thresholds

5-band (metrics.risk_band) — current model: - High ≥ 70, Moderate ≥ 50, Elevated ≥ 30, Guarded ≥ 15, else Low. Tri-band (risk_level) used on preview tiles (helper-based): - High ≥ 120, Moderate ≥ 60, else Low. Keep thresholds centralized in service helpers to avoid drift.

ASCII ERD


assessment_types (1) ──< assessments (N)
assessment_sites (0..1)
assessment_plot_types (N)
assessment_plot_type_responses (by step_slug)
assessment_units (N)
assessment_unit_responses (by step_slug)
assessment_unit_results (1 latest)
assessment_reports (0..N unit reports)
assessments ──< assessment_reports (plot-level rows with plot_type_id & kind='plot')

Where does an individual property live?

assessment_units. Each row = one property (Unit). If part of a plot, plot_type_id is set; for single/multi, plot_type_id is null.

Flow A: Plots (developments)

1) Create assessment mode 'plots' → POST /api/assessments/{type} {mode:'plots'} → {uuid} 2) Add site & plots (bulk) → POST /api/assessments/{uuid}/plot-types {site, plots:[{ref,identifiers}]} - Creates plot_types and units (meta.plot_identifier, address seeded via assessment_sites). 3) Answer plot steps → PUT /api/assessments/{uuid}/plot-types/{id}/responses/{step} 4) Process (compute+PDFs) → POST /api/assessments/{uuid}/process (ProcessAssessmentPlotsJob) - For each plot_type_id: computeAndPersistUnit for its units, then generatePlotPdf → assessment_reports(kind='plot'). 5) Preview reads cached/job snapshot or assessment_reports.meta.

Flow B: Single / Multi (individual properties)

1) Create assessment mode 'single'|'multi' → POST /api/assessments/{type} {mode:'single'} → returns {uuid, unit_uuid} 2) Manage properties: - List units GET /api/assessments/{uuid}/units - Add unit POST /api/assessments/{uuid}/units {label,address,postcode} (seeds property step) - Update unit PATCH /api/assessments/{uuid}/units/{unitUuid} (keeps property step in sync) 3) Answer steps per unit → PUT /api/assessments/{uuid}/units/{unitUuid}/responses/{step} 4) Compute & PDF per unit → POST /api/assessments/{uuid}/units/{unitUuid}/compute, /report → assessment_reports(kind='unit').

Controllers (who does what)

AssessmentController@store - Creates assessment; creates first unit unless mode='plots'. AssessmentUnitController - index → list with progress/next_step (reads responses). - store → create unit; optionally seed 'property' step from address/postcode. - update → updates meta/label/status; syncs 'property' step if address/postcode provided. - destroy/duplicate. AssessmentPlotTypeController - index/store (single or bulk), per-plot responses, compute/generatePdfs (older), download (new). ProcessAssessmentPlotsJob - Groups by plot_type_id, compute each unit, generatePlotPdf, cache progress, writes assessment_reports (plot). ComputePersistenceService - gatherAnswers(unit): plot defaults + unit overrides. - compute(answers): returns summary/metrics/advisories. - computeAndPersistUnit: writes unit_results and mirrors metrics/risk_level onto units. - generatePlotPdf: cover + per-unit, uploads, writes assessment_reports (plot).

API quick sheet

Assessments: - POST /api/assessments/{typeSlug} {mode:'single'|'multi'|'plots'} Units (individual properties): - GET /api/assessments/{uuid}/units - POST /api/assessments/{uuid}/units - PATCH/DELETE /api/assessments/{uuid}/units/{unitUuid} Unit responses: - PUT/GET /api/assessments/{uuid}/units/{unitUuid}/responses/{step} Plots: - GET /api/assessments/{uuid}/plot-types - POST /api/assessments/{uuid}/plot-types (single or bulk) Plot responses: - PUT/GET /api/assessments/{uuid}/plot-types/{id}/responses/{step} Processing: - POST /api/assessments/{uuid}/process - GET /api/assessments/{uuid}/process/{token} Reports: - GET /api/assessments/{uuid}/plot-types/{id}/pdf?store=1 - POST /api/assessments/{uuid}/units/{unitUuid}/report

Common pitfalls

- assessment_unit_responses has NO assessment_id; query by assessment_unit_id + step_slug. - Single flow must not call plot-type routes; use unit routes for steps. - Always set uuid when creating units programmatically. - Dompdf remote images: enable remote & public/signed HTTPS URLs. - Header/footer overlap: increase @page margins and reserve fixed header height. - Action totals on preview: prefer distinct counts or averages per property to avoid scary totals.

Handy queries

```
-- Units for a plot with identifiers
select id, meta->>'.plot_identifier'ident, meta -> >'.label' label
from assessment_units
where assessment_id = ? and plot_type_id = ?;

-- Unit responses (no assessment_id here)
select step_slug, json_pretty(data)
from assessment_unit_responses
where assessment_unit_id = ?;

-- Plot defaults saved
select step_slug, json_pretty(data)
from assessment_plot_type_responses
where assessment_plot_type_id = ?;

-- Latest persisted result
select json_pretty(metrics), json_pretty(advisories)
from assessment_unit_results
where assessment_unit_id = ?;

-- Reports saved
select kind, plot_type_id, unit_id, path, url, json_pretty(meta)
from assessment_reports
where assessment_id = ?;
```