
Epiphany

QUALITY METRIC CATALOG

A companion to the Core Specification

Version 0.3.0 — Phase 3 (the normative metric set: formal definitions, normalization, weights, tier thresholds, profile registry; `spacing_distortion` scoped to rhythmic columns; `vertical_density_penalty` counted per realization)

Normative for the metrics and thresholds it defines

Contents

1	About This Companion	2
1.1	Relationship to the Core Specification	3
1.2	Conformance	3
2	The Metric Model	5
2.1	Diagnostic Status	5
2.2	Determinism and Numeric Agreement	6
2.3	The Normative Metric Set and <code>QualityMetricKind</code>	6
2.4	The Measurement Domain	7
2.5	The Vacuous-Geometry Rule	8
2.6	Normalization Form	9
3	The Nine Normative Metrics	10
3.1	<code>collision_penalty</code>	10
3.2	<code>spacing_distortion</code>	11
3.3	<code>slur_shape_penalty</code>	12
3.4	<code>beam_slope_penalty</code>	13
3.5	<code>vertical_density_penalty</code>	13
3.6	<code>system_break_penalty</code>	15
3.7	<code>page_fill_efficiency</code>	16
3.8	<code>casting_off_quality</code>	16
3.9	<code>symbol_density_uniformity</code>	17
4	Default Tie-Breaking Weights	18
5	Per-Tier Metric Thresholds	19
5.1	The Default Threshold Table	19
5.2	The Advanced Tier	20
5.3	The <code>QualityFloorApproached</code> Warning	20
5.4	Standard-Tier Constraint Families	20
6	The Registered Profile Catalog	22
7	Revision History	24

About This Companion

The *Quality Metric Catalog* is a companion to the Epiphany Core Specification. It fulfils the delegation of the core specification’s COMPANION SPECIFICATIONS appendix — the section labeled `sec:deferred:companions` — which charters this document to deliver “per-metric normalization functions mapping raw measurements to `NormalizedMetric` values, default tie-breaking weights, per-tier metric thresholds, and the formal definition of each quality metric in the normative metric set.”

This companion (v0.3.0) delivers all four chartered items, plus two small registries the core specification names but defers here:

- the formal definition of each of the **nine normative metric axes** — the measured phenomenon, the raw measurement over resolved layout geometry, and the normalization function with its pinned anchor constant (Chapter 3);
- the **default tie-breaking weights** (Chapter 4);
- the **per-tier metric thresholds** for the Minimal and Standard conformance tiers, the Advanced-tier extension rule, and the `QualityFloorApproached` warning trigger (Chapter 5);
- the `QualityMetricKind` enumeration, which the core specification references (as the payload of the `QualityFloorApproached` solver warning) but never lists (Section 2.3);
- the **registered SolverProfile catalog**, which the core specification’s vocabulary appendix explicitly defers to this companion (Chapter 6);
- the **Standard-tier constraint family** declaration, which the core specification’s Standard-tier requirement points at this companion (Section 5.4).

This document does *not* cover:

- the reference suite’s test scores, per-tier entry inclusion, and any per-entry threshold overrides — those are the *Reference Suite* companion’s;
- performance conformance (edit traces, frame budgets) — the *Performance Reference Suite* companion’s;
- the reference solving algorithm — the non-normative *Reference Algorithm* companion’s.

1.1 Relationship to the Core Specification

This companion does not restate the metric framework; it *references* it. The framework — the `NormalizedMetric` validity rules (finite, in $[0.0, 1.0]$, lower is better), the `QualityMetricVector` field set, extension metrics, the `TieBreakingWeights` structure, the Pareto-frontier design target, the conformance-tier ladder, and the suite-based conformance model — is the core specification’s Chapter 9 (THE CONSTRAINT SOLVER INTERFACE, the `ch:solver` chapter), in particular its QUALITY METRICS, CONFORMANCE TIERS, and CONFORMANCE: THE REFERENCE SUITE sections (`sec:solver:quality`, `sec:solver:tiers`, `sec:solver:conformance`).

Two core requirements bind this document into the conformance story:

- The core QUALITY METRICS normalization requirement: “Per-metric normalization functions (mapping raw measurements to $[0.0, 1.0]$) are specified in the Quality Metric Catalog companion document. Implementations **MUST** use the catalog’s normalization; arbitrary normalization is non-conforming.” Chapter 3 is that normalization.
- The core tie-breaking requirement: “Tie-breaking weights **MUST** have normative defaults specified in the Quality Metric Catalog.” Chapter 4 is those defaults.

Where this document and a ratified core requirement disagree, **the core requirement governs** and the discrepancy is a defect in this document. Graph types, the layout IR pipeline (`LogicalLayoutIR` $\rightarrow$ `ConstrainedLayoutIR` $\rightarrow$ `ResolvedLayoutIR`), the spring-slot and vertical-band models, and the built-in `LayoutConstraint` kinds are the core specification’s Chapter 7 (`ch:layout-ir`); this document’s formulas range over those structures without redefining them.

RATIONALE

Versioning. This companion is versioned independently of the core specification (independent semver), like the Operation Catalog and the Binary Format companions. Metric definitions and thresholds are expected to be tuned on a faster cadence than the solver framework: threshold revisions informed by reference-suite experience are MINOR revisions here and require no core-spec change, while a change to the metric *field set* (a new normative axis) is a core-spec change first, mirrored here.

1.2 Conformance

The metric definitions, normalization functions, default weights, threshold tables, and profile registry in this document are **normative**. A solver that reports a `QualityMetricVector` computed by any function other than the ones defined here is non-conforming, per the core QUALITY METRICS requirement quoted above.

Conformance *claims* are evaluated on the Reference Suite companion’s entry set: a solver claiming tier T must keep every normative metric within tier T ’s threshold (Chapter 5) on every suite entry required at tier T . This document defines *what is measured and how much is tolerable*; the Reference Suite companion defines *on which scores*.

Two boundaries of that claim, developed in Chapter 2:

- Metric values are *diagnostic*, never canonical state (Section 2.1). No byte of canonical document state depends on them.
- Numeric agreement across implementations is *not* required (Section 2.2). The cross-implementation contract is threshold conformance, not value equality.

The Metric Model

2.1 Diagnostic Status

The quality metric vector rides on the `SolveReport` (core specification Chapter 9, `THE SOLVER REPORT: SolveReport.metric_vector`). It describes the layout; it is not part of the layout. The solver’s canonical output — `ResolvedLayoutIR` — carries no metric field, and the core specification’s observational-equivalence rule is stated over `ResolvedLayoutIR` bytes alone.

REQUIREMENT 2.1

Quality metrics are **diagnostic output**, never canonical state.

- A `QualityMetricVector` appears only on the `SolveReport`. The canonical serialized form of `ResolvedLayoutIR` **MUST NOT** contain quality-metric values, and a `NormalizedMetric` value **MUST NOT** enter canonical document bytes by any other path.
- Two solves whose `ResolvedLayoutIR` values are byte-identical under canonical serialization are observationally equivalent regardless of their metric vectors. A metric value **MUST NOT** be an input to any canonical-state decision.

RATIONALE

Keeping metrics off the canonical path is what makes them safely improvable. A solver revision that measures more honestly (or a catalog revision that tunes a formula) changes reports, warnings, and conformance verdicts — but not one byte of any document. The reference implementation already has this shape: `ResolvedLayoutIR` has no metric field, the `SolveReport` is never serialized, and no consumer reads the vector to make a state decision.

2.2 Determinism and Numeric Agreement

REQUIREMENT 2.2

Within one implementation version, metric computation **MUST** be deterministic: identical solve inputs (the same `ConstrainedLayoutIR`, configuration, and declared page geometry) **MUST** yield bitwise-identical `QualityMetricVector` values.

Across implementations (and across versions of one implementation), numeric agreement is **not** required. Two conforming solvers **MAY** report different metric values for the same score; the cross-implementation contract is the core specification’s four suite-conformance conditions — in particular, that every metric is within the claimed tier’s threshold on every required suite entry — not value equality.

Metric values are ordinary IEEE 754 `f64` values subject to the core `NormalizedMetric` validity rules (finite, in `[0.0, 1.0]`). This document imposes no additional quantization, rounding, or evaluation- order discipline on their computation.

RATIONALE

Different conforming solvers legitimately produce different layouts, so their metric values differ even under identical formulas; demanding numeric agreement would smuggle cross-implementation layout equality in through the diagnostics. Within-implementation determinism, by contrast, is load- bearing: reproducible reports are what make threshold conformance testable and regressions attributable.

2.3 The Normative Metric Set and `QualityMetricKind`

The nine normative metric axes are the nine non-extension fields of the core specification’s `QualityMetricVector`. The core references a `QualityMetricKind` enumeration (the payload of `SolverWarningKind::QualityFloorApproached`) without listing it; this catalog pins it.

REQUIREMENT 2.3

The `QualityMetricKind` enumeration is exactly:

```
pub enum QualityMetricKind {
    Collision,
    Spacing,
    SlurShape,
    BeamSlope,
    VerticalDensity,
    SystemBreak,
    PageFill,
    CastingOff,
    SymbolDensity,
}
```

Each kind names exactly one `QualityMetricVector` field and exactly one

`TieBreakingWeights` field, per Table 2.1. Extension metrics are not `QualityMetricKind` values; they are identified by `ExtensionMetricId`.

Kind	Vector field	Weight field
Collision	<code>collision_penalty</code>	<code>collision</code>
Spacing	<code>spacing_distortion</code>	<code>spacing</code>
SlurShape	<code>slur_shape_penalty</code>	<code>slur_shape</code>
BeamSlope	<code>beam_slope_penalty</code>	<code>beam_slope</code>
VerticalDensity	<code>vertical_density_penalty</code>	<code>vertical_density</code>
SystemBreak	<code>system_break_penalty</code>	<code>system_break</code>
PageFill	<code>page_fill_efficiency</code>	<code>page_fill</code>
CastingOff	<code>casting_off_quality</code>	<code>casting_off</code>
SymbolDensity	<code>symbol_density_uniformity</code>	<code>symbol_density</code>

Table 2.1: The nine normative axes: kind, vector field, tie-breaking weight.

A naming caution: three field names read as higher-is-better words —

`page_fill_efficiency`, `casting_off_quality`, `symbol_density_uniformity`

— but they are not. The core specification fixes the orientation of *every* normative metric (0.0 best, 1.0 worst tolerable), and the definitions in Chapter 3 follow it: each of the three measures a *deficiency* (unfilled page area, uneven casting-off, uneven density).

2.4 The Measurement Domain

Every raw measurement in Chapter 3 is a deterministic function of three inputs, all of which exist at the moment the solver assembles its `SolveReport`:

1. the solve’s resolved output L (a `ResolvedLayoutIR`: positioned glyphs with bounding boxes, strokes, and the page/system tree);
2. the solve’s constrained input C (a `ConstrainedLayoutIR`: horizontal spring slots, vertical bands, declared constraints);
3. the declared page geometry the solve was configured with: the content width W and content height H , in staff spaces. (Schema major 1 defines `Canvas.layout_defaults` and its type `CanvasLayoutDefaults`, P12-I7; the reference implementation’s code graph home lands in a later phase, so until then the geometry is a solver parameter, and the Reference Suite companion requires each suite entry to declare it.)

Notation used throughout Chapter 3:

- G is the set of resolved glyphs of L . For $g \in G$, the *ink box* $B(g) = [l_g, r_g] \times [b_g, t_g]$ is the glyph’s bounding box translated to its resolved position. Strokes (staff lines, ledger lines, stems, barline strokes) are not members of G .
- $sys(g)$ is the system that positioned g under the solve’s casting-off; every system belongs to exactly one region, and every page carries an ordered list of systems. A glyph positioned by no system belongs to no collision pair and to no per-system aggregate.

- $\text{slot}(g)$ is the horizontal spring slot of g 's source glyph in C — the musical time column that groups a chord's noteheads with their accidentals, dots, and same-column symbols.
- For a system s : its *columns* are the ascending sequence of distinct resolved baseline x -coordinates $x_1^s < \dots < x_{m_s}^s$ of the glyph-bearing slots realized in s ; its *advances* are $a_i^s = x_{i+1}^s - x_i^s$ for $i = 1, \dots, m_s - 1$ (equivalently, the spacing pass's per-slot advances); w_s is the width of s 's content extent (the horizontal span of the ink boxes assigned to s); n_s is the number of glyphs assigned to s .
- $\text{CV}(v_1, \dots, v_k)$, defined for $k \geq 2$ with mean > 0 , is the population standard deviation divided by the arithmetic mean.
- The arithmetic mean over an *empty* index set is defined as 0 (this is the vacuous-geometry rule of Section 2.5 in aggregate form).

Because numeric agreement across implementations is not required (Requirement 2.2), a formula may reference the solve's *own* internal assignments — which glyph landed in which system, which columns a system realizes — without threatening conformance: the assignments are deterministic within an implementation version, which is all the metric contract needs. No formula in this catalog requires an optical-spacing model, font metrics beyond glyph bounding boxes, or any geometry class the layout pipeline does not produce.

2.5 The Vacuous-Geometry Rule

Each axis in Chapter 3 names its *contributing units*: the glyph pairs, systems, pages, gaps, slurs, or beams the raw measurement ranges over. A layout may simply not contain a metric's geometry class — no drawn slurs, no beams, a single system, a single page.

REQUIREMENT 2.4

When a normative metric's contributing-unit set is empty for a given layout, the metric **MUST** evaluate to exactly 0.0: where there is nothing to penalize, the penalty is zero. In particular:

- a layout containing no drawn slur geometry has `slur_shape_penalty = 0.0`;
- a layout containing no drawn beam geometry has `beam_slope_penalty = 0.0`;
- a region cast onto a single system contributes no units to `system_break_penalty`, `casting_off_quality`, or `symbol_density_uniformity`, and a single-page layout contributes no units to `page_fill_efficiency` — each axis degenerates exactly as its per-axis definition states;
- a solve configured without positive finite content bounds (W or H) has an empty contributing set for every axis defined over that bound.

An implementation **MUST NOT** report a sentinel (such as 1.0) for a metric whose contributing-unit set is empty. The all-worst placeholder vector remains correct only for a solver that *computes no metrics at all* and claims no conformance tier (the core's `Stub` tier).

OPEN QUESTION

The notated-but-unrendered honesty edge. A score whose *source* notates geometry a solver does not draw scores vacuous-0.0 on that axis under this rule — the axis sees nothing drawn and finds nothing to penalize, even though the output is arguably *worse* than a badly-drawn one. The metric axes evaluate the geometry the solver produced, and *rendering completeness* — whether notated content is realized at all — is governed by constraint families and visual acceptance testing, not by the quality metrics. Should a future revision instead score notated-but-unrendered geometry classes at the worst value, so that the metric vector cannot flatter an incomplete renderer? Resolving this requires a normative definition of “notated content that demands drawn geometry,” which does not exist yet. *Slurs no longer instance this edge* (they render and are measured, schema-major-2); it persists for still-logical-only classes such as beams.

2.6 Normalization Form

Every normative axis uses the same one-parameter normalization shape, so that anchors — not curve families — are the entire tuning surface.

REQUIREMENT 2.5

Each normative metric defines a raw measurement $raw \geq 0$ (dimensionless, per its axis definition) and a pinned anchor constant $R_{\text{worst}} > 0$. The normalized value is the clamped-linear map

$$n = \min\left(1, \frac{raw}{R_{\text{worst}}}\right),$$

so that $raw = 0$ (the ideal) normalizes to 0.0 and $raw \geq R_{\text{worst}}$ (the worst-tolerable anchor and beyond) normalizes to 1.0. Implementations **MUST** use the per-axis raw measurements and anchors of Chapter 3 exactly; per the core specification, arbitrary normalization is non-conforming. Extension metrics **MAY** use other normalization shapes but **MUST NOT** change orientation or range.

The Nine Normative Metrics

Each section below defines one axis under a fixed template: the *phenomenon* (what an engraver would point at), the *contributing units* (what the raw measurement ranges over — the set whose emptiness triggers Requirement 2.4), the *raw measurement*, and the *normalization anchor* with a brief justification. All lengths are in staff spaces; all raw measurements are dimensionless ratios.

Four axes measure horizontal-distribution phenomena at different granularities, and the boundaries are deliberate:

- `spacing_distortion` is *within-system* advance regularity;
- `system_break_penalty` is the *per-break* absolute cost of each chosen system break (looseness or overflow of non-final systems);
- `casting_off_quality` is *across-system* width evenness, including the final system (the stub-last-line failure);
- `symbol_density_uniformity` is *across-system* crowding evenness (equal widths can hide very different symbol densities).

3.1 collision_penalty

Phenomenon. Overlapping ink between symbols that belong to different musical time columns: a notehead striking the previous column’s accidental, a chord symbol over a barline, any cross-column ink contact. Professional engraving contains none.

REQUIREMENT 3.1

Contributing units: unordered glyph pairs $\{g, h\} \subseteq G$ with $\text{sys}(g) = \text{sys}(h)$ and $\text{slot}(g) \neq \text{slot}(h)$.

A pair *collides* when its ink boxes intersect with positive area in both axes:

$$\begin{aligned} \min(r_g, r_h) - \max(l_g, l_h) &> 0 \quad \text{and} \\ \min(t_g, t_h) - \max(b_g, b_h) &> 0. \end{aligned}$$

Edge-touching boxes do not collide. Pairs sharing a horizontal spring slot are **excluded**: a column’s internal cluster — a chord’s noteheads, their accidentals, dots, and other same-slot symbols — is arranged by the constrained stage, and its legitimate internal ink

contact is not a spacing failure of the solver. Strokes are not glyphs and join no pair: staff lines legitimately cross every notehead.

Raw measurement: with P the set of colliding pairs,

$$raw = \frac{|P|}{|G|} \quad (raw = 0 \text{ when } G = \emptyset).$$

Normalization: $R_{\text{worst}} = 0.05; n = \min(1, raw/0.05)$.

RATIONALE

The anchor says: one cross-column collision per twenty glyphs is unmistakably broken layout — the worst a report should be able to distinguish. The count is divided by the glyph population, not by the pair population, so that the measure does not vanish quadratically on large scores: a score with one collision per page stays visible. The reference pipeline evaluates overlap today only for *declared* NoCollision constraints; this axis is the full pairwise same-system sweep over ink boxes, which is new but cheap work over data the resolved layout already carries.

3.2 spacing_distortion

Phenomenon. Uneven horizontal distribution within a system: the *rhythmic* columns — those carrying notes and rests — bunched together here and stretched apart there, where the underlying spring model asked for near-uniform advances.

REQUIREMENT 3.2

Rhythmic columns. A *rhythmic column* of a system is a horizontal spring slot at least one of whose glyphs is a notehead or a rest. The clef, key-signature, and time-signature lead and the barlines are **not** rhythmic columns: their horizontal extent is notational furniture, fixed by their content rather than by rhythm.

Contributing units: systems s with at least three rhythmic columns (so at least two rhythmic advances).

Raw measurement: let $x_1^s < \dots < x_{k_s}^s$ be the reference x-positions of the system's k_s rhythmic columns in order, and $a_i^s = x_{i+1}^s - x_i^s$ the advance between consecutive rhythmic columns (a note-to-note advance spans any barline or furniture that falls between the two, so barlines are transparent, not breaks in the sequence). Then $raw_s = CV(a_1^s, \dots, a_{k_s-1}^s)$, and the axis raw value is the arithmetic mean of raw_s over contributing units.

Normalization: $R_{\text{worst}} = 1.0; n = \min(1, raw)$.

RATIONALE

A coefficient of variation of 1.0 means the typical advance deviates from the mean by the whole mean — spacing with no discernible regularity. vo.2 defines *geometric* regularity

over the *rhythmic* columns deliberately: the reference spring model’s preferred widths are uniform, so regular note/rest advances are exactly what its ideal output looks like, and the collision minima (accidental overhangs, wide columns) that legitimately perturb those advances are modest on realistic scores. The axis is scoped to rhythmic columns because a leading clef, key signature, or time signature is furniture whose width the spring model does not govern: folding the wide clef-to-first-note gap into the CV would flag a perfectly-spaced short line as distorted purely for carrying a clef — measuring furniture, not spacing.

OPEN QUESTION

Optical spacing at the Standard tier. Mature engraving spaces rhythmic columns proportionally to musical duration (with an optical correction), not uniformly; under a duration-proportional model, this axis’s ideal would be “advances proportional to the column’s duration share,” and a perfectly optically-spaced line would score *worse* than a uniform one under the vo.2 definition. When the layout pipeline gains duration-aware preferred widths, should the Standard tier redefine raw_s as deviation from the duration-proportional ideal while Minimal keeps geometric regularity? vo.2 scopes the geometric definition to rhythmic columns (removing the leading-furniture false positive on short scores) but still measures geometric, not duration-proportional, regularity.

3.3 slur_shape_penalty

Phenomenon. Badly-shaped slur arcs: flat, tape-like slurs or bulging semicircles, measured against the shallow-arc norm of engraving practice.

REQUIREMENT 3.3

Contributing units: drawn slurs in L with chord length $c > 0$, where the *chord* is the segment between the *whole* slur’s endpoints and the *apex height* $h \geq 0$ is the maximum perpendicular distance from the curve to its chord. The unit is the whole slur, not a per-system fragment: a slur that a casting-off pass splits across a system break is measured once, as the arc it was shaped to be, so a well-shaped slur that happens to break is not spuriously penalized (its fragments’ diagonal chords each read flatter than the whole).

Raw measurement: per unit, with arc ratio $\rho = h/c$,

$$raw_u = \max(0, 0.08 - \rho, \rho - 0.25),$$

i.e. the shortfall below the ideal band $[0.08, 0.25]$ or the excess above it; the axis raw value is the arithmetic mean over contributing units.

Normalization: $R_{\text{worst}} = 0.25$; $n = \min(1, raw/0.25)$.

RATIONALE

The band $[0.08, 0.25]$ brackets the shallow arcs engraving practice prefers: an arc rising less than about $1/12$ of its span reads as a straight line; one rising more than a quarter of its span begins to bulge. The anchor makes a semicircular slur ($\rho = 0.5$, $raw_u = 0.25$) exactly worst-tolerable, and a completely flat slur ($\rho = 0$, $raw_u = 0.08$) roughly a third of the way to failing.

The implementation now *draws* slurs as cubic-Bézier curves and measures this axis directly (schema-major-2 rendering; the first slur-drawing release, as the pinned definition anticipated). A layout with no drawn slur curve still evaluates to 0.0 under the vacuous-geometry rule. A tier that draws the ideal shallow arc for every slur measures 0 on this axis; a fixed-height engraver scores non-zero on slurs whose span pushes the arc ratio outside the $[0.08, 0.25]$ band (a very short or very long slur), which a duration-aware height corrects.

3.4 beam_slope_penalty

Phenomenon. Over-steep beams. Engraving practice keeps beam slants gentle regardless of the melodic interval they span.

REQUIREMENT 3.4

Contributing units: drawn beam segments in L with horizontal run $\Delta x > 0$ (endpoint-to-endpoint).

Raw measurement: per unit, with absolute slope $\sigma = |\Delta y|/\Delta x$,

$$raw_u = \max(0, \sigma - 0.25);$$

the axis raw value is the arithmetic mean over contributing units.

Normalization: $R_{\text{worst}} = 0.25$; $n = \min(1, raw/0.25)$.

RATIONALE

Slopes up to 0.25 (about 14°) are penalty-free — within the range engraving manuals tolerate for short, wide-interval beams — and the anchor places $\sigma = 0.5$ (about 27° , roughly double any published maximum) at worst-tolerable. Like the slur axis, this is pinned ahead of implementation: the vo.1 reference pipeline draws no beam geometry, so the axis evaluates to 0.0 under the vacuous-geometry rule.

3.5 vertical_density_penalty

Phenomenon. Vertical crowding or sprawl: inter-staff and inter-system gaps realized far from the spacing the band model asked for.

REQUIREMENT 3.5

Contributing units: each *realization* in L of a vertical band of C of kind `InterStaffGap` or `InterSystemGap` with preferred height $p > 0$ (a band is realized wherever the adjacent content it separates was laid out). A band realized in several systems contributes **one unit per system**, not one per band: a vertical solve sizes each system’s gaps from that system’s own content, so one band’s realized height genuinely differs from system to system. This matches how the realized inter-system gaps are already counted — one unit per adjacent system pair on a page.

Raw measurement: per unit, with $r \geq 0$ the realized vertical separation between the adjacent content extents the band separates (measured in resolved coordinates),

$$raw_u = \frac{|r - p|}{p};$$

the axis raw value is the arithmetic mean over contributing units.

Content extent means *every* primitive the adjacent band owns — glyphs, strokes, and curves alike, each attributed by its declared `vertical_band` (core spec `CONSTRAINEDLAYOUTIR`, primitive band ownership) — and not the band’s `glyph members` alone. A staff’s outermost ink is usually not a glyph: ledger lines, stems, and slurs reach past every notehead. A solver separates staves until their content clears the gap, so scoring it against its noteheads alone would charge it for the very ink it made room for.

Normalization: $R_{\text{worst}} = 1.0$; $n = \min(1, raw)$.

RATIONALE

A gap off by its own preferred size — staves twice as far apart as asked, or fully collapsed — is unambiguous vertical failure; proportional deviation makes one anchor serve both tight inter-staff gaps and wide inter-system gaps.

The vo.1 reference pipeline preserved constrained y verbatim, so realized gaps equalled preferred gaps wherever bands were realized and the axis reported its honest near-zero. The vertical spring solve has since landed (inter-staff gap renegotiation and inter-system vertical justification), and the axis now measures what it was defined to measure. The **content extent** clarification above is the lesson from that landing: the reference implementation read “content extents” as the band’s `glyph members`, because until primitive band ownership was ratified a band listed no strokes or curves to own. A correctly separated two-staff system then scored a saturated 1.0 and tripped the `STANDARD` floor warning — the metric charging the solver for the ledger and slur ink it had correctly cleared. The formula, units, anchor, and normalization are unchanged; only a non-conforming measurement was.

The inter-system half of the axis is a genuine trade-off, not a defect: the vertical justification pass deliberately stretches inter-system gaps past preferred to fill a non-final page, trading this axis against `page_fill_efficiency`. Both axes are reported; neither is wrong.

Per-realization units (0.3.0) followed from the same landing. While the reference pipeline preserved constrained y verbatim, a band had exactly one realized height and the distinction was moot. Once the inter-staff solve began sizing each system’s gaps from

that system’s own content, a band’s realized height became per-system, and counting one unit per band would let a well-solved system average away a badly spaced one. The axis is deliberately symmetric: a gap wider than preferred is sprawl exactly as a narrower one is crowding. A solver that only ever *expands* a fixed stacking — never compressing an over-wide gap back toward preferred — will therefore report honest sprawl on its slack systems. That is the axis working, not mis-measuring, and it is how the reference implementation’s expand-only solve was found: it now renegotiates in both directions. Where a solver realizes each band’s declared height exactly, the inter-staff half of this axis reads its honest zero and serves as a *self-check* on the solve rather than a judgement on the score. A conforming implementation should therefore measure the realized separation back from the geometry it produced, not from the target it aimed at: the two agree only if the solve and the bake both did what they claimed. The reference implementation reads it back from the baked output, and that is how a defect in its multi-staff cascade — one that over-separated every pair below the first — was caught.

3.6 system_break_penalty

Phenomenon. Bad break choices, one system at a time: a non-final system left loose (broken far short of the available width) or overfull (content past the content width).

REQUIREMENT 3.6

Contributing units: non-final systems — for each region, every system the casting-off produced except the region’s last — defined only when the declared content width W is finite and positive.

Raw measurement: per unit,

$$raw_s = \frac{|W - w_s|}{W},$$

penalizing looseness ($w_s < W$) and overflow ($w_s > W$) alike; the axis raw value is the arithmetic mean over contributing units.

Normalization: $R_{\text{worst}} = 0.5$; $n = \min(1, raw/0.5)$.

A region cast onto a single system contributes no units (the break axis degenerates to nothing-to-penalize, per Requirement 2.4); the final system of each region is never a unit, because a short last line is not a break failure.

RATIONALE

Non-final systems half-empty on average — or overflowing by half the content width — mark casting-off that has effectively failed, hence the 0.5 anchor. The raw quantities are exactly what the reference casting-off pass already computes: per-system content extents against the declared content width, with breaks chosen among barline candidates.

3.7 page_fill_efficiency

Phenomenon. Underfilled non-final pages: vertical white space a better page-break policy would have used. Despite the field’s name, the metric follows the fixed orientation — it measures *unfilled* fraction, so 0.0 is best.

REQUIREMENT 3.7

Contributing units: non-final pages of L , defined only when the declared content height H is finite and positive.

Raw measurement: per unit, with $span_p$ the vertical extent of page p ’s content (from the top of its first system’s content extent to the bottom of its last system’s content extent) and fill fraction $f_p = \min(1, span_p/H)$,

$$raw_p = 1 - f_p;$$

the axis raw value is the arithmetic mean over contributing units.

Normalization: $R_{\text{worst}} = 0.75$; $n = \min(1, raw/0.75)$.

A single-page layout contributes no units; the final page is never a unit, because a short last page is not a fill failure.

RATIONALE

A non-final page three-quarters empty is a page break with no plausible justification — worst-tolerable. The span-based fill fraction is computable directly from the casting-off pass’s vertical cursor walk and per-system extents, and clamping f_p at 1 keeps slight margin overshoot from producing a negative raw value.

3.8 casting_off_quality

Phenomenon. Uneven casting-off across a region’s systems taken as a whole: some lines full, others sparse — including the classic failure this axis exists to catch, a stub final system carrying one straggling measure. Despite the field’s name, 0.0 is best.

REQUIREMENT 3.8

Contributing units: regions whose casting-off produced at least two systems, each with content-extent width $w_s > 0$.

Raw measurement: per unit region R ,

$$raw_R = CV(w_s : s \in \text{systems}(R)),$$

over *all* of the region’s systems, the final system included; the axis raw value is the arithmetic mean over contributing units.

Normalization: $R_{\text{worst}} = 0.5$; $n = \min(1, raw/0.5)$.

A single-system region contributes no units.

RATIONALE

Including the final system is the deliberate difference from `system_break_penalty` (which exempts it): a lone stub last line drags the width spread up and is penalized *here*, as a global casting-off failure rather than a per-break one. The anchor: per-system widths whose standard deviation is half their mean describe a page where line lengths visibly disagree.

3.9 symbol_density_uniformity

Phenomenon. Uneven crowding across systems: one line crammed with symbols, the next sparse — even when the lines' widths agree. Despite the field's name, 0.0 is best.

REQUIREMENT 3.9

Contributing units: regions whose casting-off produced at least two systems with $w_s > 0$.

Raw measurement: per unit region R , with per-system symbol density $\rho_s = n_s/w_s$ (glyphs per staff space of content width),

$$raw_R = CV(\rho_s : s \in \text{systems}(R), w_s > 0);$$

the axis raw value is the arithmetic mean over contributing units.

Normalization: $R_{\text{worst}} = 0.5$; $n = \min(1, raw/0.5)$.

A single-system region contributes no units.

RATIONALE

Width evenness (`casting_off_quality`) and density evenness are independent failures: equal-width systems can still alternate between sixteenth-note walls and whole-note deserts when break choices ignore content weight. Density varying by half its mean across systems reads as visibly uneven engraving, hence the shared 0.5 anchor.

Default Tie-Breaking Weights

The core specification requires normative default `TieBreakingWeights`: they select among Pareto-equivalent layouts, deterministically, and are “the basis for reference-suite conformance.”

REQUIREMENT 4.1

The normative default tie-breaking weights are 1.0 for every one of the nine fields of `TieBreakingWeights`:

Weight	Default	Weight	Default
collision	1.0	system_break	1.0
spacing	1.0	page_fill	1.0
slur_shape	1.0	casting_off	1.0
beam_slope	1.0	symbol_density	1.0
vertical_density	1.0		

Implementations **MAY** let users customize weights, per the core specification; conformance evaluation on the reference suite uses these defaults.

RATIONALE

No aesthetic priority ordering among the nine axes has been ratified, and inventing one ahead of measurement experience would encode a preference no evidence supports. Uniform weights are the honest neutral default — they make tie-breaking deterministic (the core’s actual requirement) without pretending to a house style. They also bless the reference implementation’s existing `Default` for `TieBreakingWeights` (every field 1.0). Revisions of this catalog are expected to tune the defaults once reference-suite experience shows which axes dominate perceived quality.

Per-Tier Metric Thresholds

5.1 The Default Threshold Table

A tier’s threshold for an axis is the maximum permitted `NormalizedMetric` value on a reference-suite entry evaluated at that tier. The core specification fixes the relationship: Minimal-tier thresholds are relaxed relative to Standard; the Standard tier corresponds to professional engraving quality.

Axis	Minimal (max)	Standard (max)
collision_penalty	0.90	0.25
spacing_distortion	0.90	0.40
slur_shape_penalty	0.90	0.30
beam_slope_penalty	0.90	0.30
vertical_density_penalty	0.90	0.40
system_break_penalty	0.90	0.35
page_fill_efficiency	0.90	0.40
casting_off_quality	0.90	0.35
symbol_density_uniformity	0.90	0.40

Table 5.1: Default per-tier maximum `NormalizedMetric` values. Minimal is uniformly more permissive than Standard on every axis.

REQUIREMENT 5.1

The default per-tier thresholds are given by Table 5.1. A solver claiming a tier **MUST** keep every normative metric at or below the tier’s threshold on every reference-suite entry required at that tier, per the core specification’s suite-conformance conditions. The Reference Suite companion **MAY** override these defaults for individual entries; absent an override, the values of Table 5.1 govern.

RATIONALE

Minimal = 0.90 everywhere: relaxed but non-vacuous. A Minimal solver may be aesthetically mediocre — the core says so — but it must not be *pathological*, and its metric vectors must be accurate. A uniform 0.90 admits every honestly-mediocre layout while excluding two things: layouts at an axis’s worst-tolerable anchor, and the all-worst place-

holder vector of a solver that computes nothing. That second exclusion is deliberate — a solver reporting the unmeasured 1.0 placeholder cannot pass the Minimal suite, which is exactly the honest-tier discipline: measuring is part of the Minimal claim.

Standard = 0.25–0.40 per axis: professional quality. Collisions get the tightest bound (0.25: at most one cross-column collision per eighty glyphs) because they are the most jarring single defect. The break-family axes (0.35) sit slightly tighter than the distribution and vertical axes (0.40), whose v0.1 definitions are coarser proxies (geometric spacing regularity; a not-yet-solved vertical dimension). Slurs and beams (0.30) allow modest shape deviation across a piece. All values are round v0.1 defaults chosen to be defensible, not optimal; the tuning open question below owns their evolution.

5.2 The Advanced Tier

REQUIREMENT 5.2

The Advanced tier imposes the Standard-tier thresholds of Table 5.1 on the nine normative axes, *plus* per-extension thresholds on extension metrics: a registered extension whose layout requirements are part of the Advanced reference suite **MUST** declare, in its extension declaration, a maximum `NormalizedMetric` value for each extension metric it contributes, and an Advanced-tier solver **MUST** meet each declared threshold on every Advanced suite entry that exercises that extension. An extension metric with no declared threshold imposes no Advanced-tier obligation.

5.3 The `QualityFloorApproached` Warning

The core specification gives `SolverWarningKind` a `QualityFloorApproached` variant carrying a `QualityMetricKind` payload, without defining its trigger. This catalog pins it.

REQUIREMENT 5.3

A solver **SHOULD** emit a `QualityFloorApproached` warning for metric kind k when the computed value of k 's axis exceeds 0.8 times the applicable threshold for that axis. The applicable threshold is the one selected by the solve's `SolverProfile` (Chapter 6); the warning fraction is pinned at 0.8 exactly. The warning is diagnostic: emitting it does not change the solve's status, and a value *over* the threshold still warns (it exceeds 0.8 of it a fortiori) — threshold *enforcement* exists only in reference-suite evaluation, not in ordinary solves.

5.4 Standard-Tier Constraint Families

The core specification's Standard-tier requirement obliges a Standard solver to “support every Standard-tier constraint family declared in the Quality Metric Catalog.” This section is that declaration.

REQUIREMENT 5.4

The Standard-tier constraint families are the core specification’s built-in layout-constraint surface (Chapter 7, `CONSTRAINEDLAYOUTIR`):

- the **spring families**: horizontal spring slots and vertical bands, with their min/preferred/max and stretch/compress parameters;
- the five built-in `LayoutConstraint` kinds: `NoCollision`, `Align`, `PositionWithin`, `SystemBreakAt`, and `PageBreakAt` (both `Hard` and `Soft` break kinds).

These same families constitute “the standard constraint families” of the core’s Minimal-tier requirement: Minimal and Standard support the same family set and differ in metric thresholds and incremental-solving obligations, not in constraint vocabulary.

`LayoutConstraint::Registered` (extension-contributed) families are per-extension obligations of the Advanced tier only.

OPEN QUESTION

Threshold tuning. Every number in Table 5.1 and every anchor constant in Chapter 3 is a v0.1 default pinned ahead of measurement experience: no implementation has yet reported real vectors across the reference suite. Once the reference implementation computes real metrics on the v0.1 entry set, are the Standard columns achievable-but-meaningful (neither trivially passed nor unreachable), and do any anchors need rescaling? Threshold and anchor revisions are MINOR versions of this catalog and are expected.

The Registered Profile Catalog

The core specification’s vocabulary appendix defines `SolverProfile` as a registered profile identifier that “selects the solver’s hard-constraint set, normalized-metric thresholds, tie-breaking weights, and active extension catalog,” and defers the registry to this companion.

REQUIREMENT 6.1

The registered `SolverProfile` catalog is exactly three profiles: `Draft`, `Standard`, and `Publication`. Their selections:

Profile	Constraint families	Threshold column	Weights	Extensions
<code>Draft</code>	Standard-tier set	Minimal	defaults	none required
<code>Standard</code>	Standard-tier set	Standard	defaults	none required
<code>Publication</code>	Standard-tier set	Standard	defaults	none required

- *Constraint families*: all three profiles activate the Standard-tier constraint families of Requirement 5.4; hard constraints are never traded away by any profile (the core’s hard-constraints-are-inviolable rule).
- *Threshold column*: the column of Table 5.1 the profile selects — the thresholds against which Requirement 5.3’s warning fraction is evaluated during ordinary solves. `Draft` selects the Minimal column (few warnings, fast iteration); `Standard` and `Publication` select the Standard column. As of v0.2 no column tighter than Standard is ratified; `Publication` is registered now so that documents and configurations can name it, and a future revision **MAY** give it a tighter column without a schema change.
- *Weights*: all three profiles use the default tie-breaking weights of Requirement 4.1.
- *Extensions*: no profile requires an active extension catalog; extensions activate by document declaration, not by profile.

`Standard` is the default profile.

RATIONALE

Profiles are configuration; tiers are claims. A `SolverProfile` is a runtime input (`SolverConfig.profile`) that any solver may be asked to run under; a conformance tier is a claim about the solver evaluated on the reference suite. The two meet in exactly one place: the profile’s threshold column determines which thresholds the solver’s own

QualityFloorApproached diagnostics reference during ordinary solves. Suite evaluation at a claimed tier always uses that *tier's* column, whatever profile the solver runs under day to day. The three-profile registry matches the reference implementation's existing SolverProfile enum (Draft / Standard / Publication, default Standard) so that registration blesses shipped reality rather than inventing a parallel one.

Revision History

Date	Section	Change
July 2026	21, All	0.1.0 — Initial companion: pins the diagnostic-only status of quality metrics, within-implementation determinism without cross-implementation numeric agreement, the <code>QualityMetricKind</code> enumeration, the measurement domain, and the vacuous-geometry rule; defines all nine normative axes (phenomenon, raw measurement over resolved geometry, clamped-linear normalization with pinned anchors); sets the default tie-breaking weights (all 1.0); establishes the Minimal/Standard threshold table, the Advanced extension rule, the <code>QualityFloorApproached</code> trigger ($0.8\times$ threshold), and the Standard-tier constraint family declaration; registers the Draft/Standard/Publication profile catalog. Open questions: the notated-but-unrendered honesty edge, optical spacing at the Standard tier, threshold tuning pending reference-suite experience.
July 2026	21, <code>spacing_distortion</code>	0.1.0 — Scope the <code>spacing_distortion</code> raw measurement to the system's <i>rhythmic</i> columns (slots bearing a notehead or rest), excluding the clef / key-signature / time-signature lead and treating barlines transparently (a note-to-note advance spans them). Resolves the reference-suite false positive in which a short healthy line's wide clef-to-first-note gap inflated the CV above the Standard warning floor (measured 0.36–0.41 on the three- to eight-column entries $\rightarrow$ 0.08–0.22, below the floor) without weakening the axis on real spacing irregularity. Normalization anchor, orientation, range, thresholds, and the eight other axes are unchanged; the optical-spacing open question (duration-proportional spacing) stays open. Batch item P12-I12.

Date	Section	Change
July 2026	21, <code>vertical_density_penalty</code>	Count one contributing unit per realization of an <code>InterStaffGap</code> band rather than one per band , matching how realized inter-system gaps were already counted. The inter-staff vertical solve sizes each system’s gaps from that system’s own content, so a band’s realized height is per-system; one unit per band let a well-solved system average away a badly spaced one. Also <i>clarifies</i> (no semantic change) that the “content extents” the raw measurement compares are every primitive the adjacent band owns — glyphs, strokes, and curves, each attributed by its declared <code>vertical_band</code> — and not the band’s <code>glyph</code> members, a reading that was arguably unimplementable before primitive band ownership was ratified. Raw formula, normalization anchor, orientation, range, thresholds, and the eight other axes are unchanged. The reference implementation’s non-conforming glyph-only measurement is fixed alongside.