CIE v2.3.1

DEVELOPER BUILD PACK

8 Technical Specifications for PHP + Python Implementation

February 2026 | CONFIDENTIAL | Hand to Dev Lead on Day 1

Companion to: CIE v2.3.1 Enforcement Edition + Developer Technical Specification

Build Pack Contents

This pack contains the 8 missing technical documents identified by cross-engine review. Together with the CIE v2.3.1 Enforcement + Dev Spec, this gives your PHP and Python teams everything they need to build without asking clarification questions.

Doc	Name	What It Unlocks	Audience
1	Canonical Database Schema	Tables, fields, types, FKs, enums — the data truth	All devs
2	OpenAPI Specification (YAML)	Importable API contract for Swagger/Postman	Backend devs
3	RBAC & Permissions Matrix	Who can touch what — enforces 'no override'	Backend + DevOps
4	Test Fixtures & Golden Dataset	30 SKUs with expected gate outcomes for automated testing	QA + all devs
5	Sequence Diagrams	SKU lifecycle, publish flow, audit job, decay loop	All devs + architects
6	Title Construction Engine	Pseudocode for intent-first title generation per product class	PHP + content devs
7	Immutable Audit Logging Spec	Every change tracked — who, when, before/after	Backend + DevOps
8	Channel Output Mapping	CIE fields → Amazon, Google Shopping, own site feeds	Integration devs

Completion test: Dev team should need <10 clarification questions total across all 8 docs.

Document 1: Canonical Database Schema

This is the single source of truth for all data structures. Every field name, type, constraint, and relationship is defined here. Devs must implement these exactly — no creative interpretation.

1.1 Entity Relationship Overview

```
┌─────────────────┐        1:N         ┌───────────────────────┐
│ cluster_master  │ ──────────────────→│ sku_master            │
│                 │                    │                       │
│ cluster_id (PK) │                    │ sku_id (PK)           │
│ category        │                    │ cluster_id (FK)       │
│ intent_statement│                    │ tier (enum)           │
│ intent_vector   │                    │ primary_intent (FK)   │
└─────────────────┘                    └───────────┬───────────┘
                                                   │
┌─────────────────┐                    ┌───────────┴───────────┐
│ intent_taxonomy │                    │ sku_secondary_intents │
│                 │←───────────────────│                       │
│ intent_id (PK)  │      FK            │ sku_id (FK)           │
│ intent_key      │                    │ intent_id (FK)        │
│ tier_access[]   │                    │ ordinal               │
└─────────────────┘                    └───────────────────────┘


┌─────────────────┐                    ┌───────────────────────┐
│ sku_content     │                    │ sku_gate_status       │
│                 │                    │                       │
│ sku_id (FK, PK) │                    │ sku_id (FK)           │
│ title           │                    │ gate_code (enum)      │
│ answer_block    │                    │ status (pass/fail)    │
│ best_for[]      │                    │ error_code            │
│ not_for[]       │                    │ checked_at            │
│ expert_authority│                    └───────────────────────┘
│ wikidata_uri    │
└─────────────────┘                    ┌───────────────────────┐
                                       │ channel_readiness     │
┌─────────────────┐                    │                       │
│ sku_tier_history│                    │ sku_id (FK)           │
│                 │                    │ channel (enum)        │
│ sku_id (FK)     │                    │ score (0-100)         │
│ old_tier        │                    │ component_scores{}    │
│ new_tier        │                    │ computed_at           │
│ changed_at      │                    └───────────────────────┘
│ changed_by      │
│ reason          │                    ┌───────────────────────┐
└─────────────────┘                    │ ai_audit_results      │
                                       │                       │
┌─────────────────┐                    │ audit_run_id (FK)     │
│ ai_audit_runs   │────────────────────│ question_id (FK)      │
│                 │       1:N          │ engine (enum)         │
│ run_id (PK)     │                    │ score (0-3)           │
│ category        │                    │ response_snippet      │
│ run_date        │                    │ cited_sku_id          │
│ status          │                    └───────────────────────┘
└─────────────────┘
                                       ┌───────────────────────┐
┌─────────────────┐                    │ audit_log (IMMUTABLE) │
│ ai_golden_queries                    │                       │
│                 │                    │ log_id (PK, UUID)     │
│ question_id (PK)│                    │ entity_type           │
│ category        │                    │ entity_id             │
│ question_text   │                    │ action                │
│ intent_type (FK)│                    │ field_name            │
│ query_family    │                    │ old_value             │
│ target_skus[]   │                    │ new_value             │
│ locked_until    │                    │ actor_id              │
```

```
└──────────────────┘                │ actor_role              │
                                    │ timestamp               │
                                    │ ip_address              │
┌──────────────────┐                 └────────────────────────┘
│ material_wikidata│
│                  │
│ material_id (PK) │
│ material_name    │
│ wikidata_qid     │
│ wikidata_uri     │
│ ai_signal        │
└──────────────────┘
```

1.2 Table Definitions

cluster_master

One row per semantic cluster. Owned by SEO Governor. Changes require Governor sign-off and trigger audit log entry.

Field	Type	Nullable	Default	Index	Constraint / Notes
cluster_id	VARCHAR(50)	NO	-	PK	Format: CLU-{CATEGORY}-{TYPE}-{SUBTYPE}. Immutable after creation.
category	ENUM	NO	-	IDX	Values: cables, lampshades, bulbs, pendants, floor_lamps, ceiling_lights, accessories
intent_statement	VARCHAR(500)	NO	-	-	Human-readable intent. e.g. 'Connect and power a pendant light safely and stylishly'
intent_vector	VECTOR(1536)	NO	-	-	Embedding of intent_statement. Recomputed when statement changes. Model: text-embedding-3-small
is_active	BOOLEAN	NO	true	-	Soft delete. Inactive clusters block new SKU assignments.
created_at	TIMESTAMP	NO	NOW()	-	Immutable
updated_at	TIMESTAMP	NO	NOW()	-	Auto-updated on change

sku_master

One row per SKU. Core product record with tier, cluster assignment, and status.

Field	Type	Null	Default	Index	Constraint / Notes
sku_id	VARCHAR(50)	NO	-	PK	Format: SKU-{CATEGORY}-{SEQ}. Immutable.

cluster_id	VARCHAR(50)	NO	-	FK,IDX	FK→cluster_master.cluster_id. Must exist in master list. G1 gate.
tier	ENUM	NO	-	IDX	Values: hero, support, harvest, kill. G6 gate. Set by ERP sync.
primary_intent_id	SMALLINT	NO	-	FK	FK→intent_taxonomy.intent_id. G2 gate. Must be in tier_access.
status	ENUM	NO	draft	IDX	Values: draft, ready, published, archived. Transition to 'published' requires ALL gates pass.
erp_margin_pct	DECIMAL(5,2)	YES	NULL	-	From ERP sync. Used in tier calculation.
erp_cppc	DECIMAL(8,4)	YES	NULL	-	Cost per product click from ERP/Ad platform.
erp_velocity_90d	INTEGER	YES	NULL	-	90-day unit sales from ERP.
erp_return_rate_pct	DECIMAL(5,2)	YES	NULL	-	Return rate from ERP.
commercial_score	DECIMAL(8,4)	YES	NULL	IDX	Computed: (margin*0.4)+(1/cppc*0.25)+(vel*0.2)+((1-ret)*0.15)
decay_status	ENUM	NO	none	IDX	Values: none, yellow_flag, alert, auto_brief, escalated. Citation decay tracker.
decay_consecutive_zeros	SMALLINT	NO	0	-	Count of consecutive weeks with score=0 on any tracked question.
created_at	TIMESTAMP	NO	NOW()	-	Immutable
updated_at	TIMESTAMP	NO	NOW()	-	Auto-updated

intent_taxonomy

Exactly 9 rows. Locked. Changes require quarterly review.

Field	Type	Example	Null	Index	Notes
intent_id	SMALLINT	1	NO	PK	1-9. Immutable.
intent_key	VARCHAR(30)	problem_solving	NO	UNQ	Snake_case. Used in API payloads.
label	VARCHAR(50)	Problem-Solving	NO	-	Display label in CMS UI.
definition	VARCHAR(200)	User has a problem...	NO	-	Shown as tooltip in CMS.
tier_access	JSON ARRAY	["hero","support","harvest"]	NO	-	Which tiers can use this intent. G6.1 enforcement.

sku_secondary_intents

Junction table. 1-3 rows per SKU. G3 gate enforces count + uniqueness.

Field	Type	Nullable	Index	Constraint
sku_id	VARCHAR(50)	NO	PK(composite),FK	FK→sku_master.sku_id
intent_id	SMALLINT	NO	PK(composite),FK	FK→intent_taxonomy.intent_id. Must differ from sku_master.primary_intent_id.
ordinal	SMALLINT	NO	-	1-3. Display order.

CONSTRAINT: Max rows per sku_id

hero: max 3 secondary intents

support: max 2 secondary intents

harvest: max 1 secondary intent

kill: 0 rows allowed (insert blocked at API level)

sku_content

One row per SKU. All content fields subject to gate validation.

Field	Type	Null	Gate	Constraint
sku_id	VARCHAR(50)	NO	-	PK, FK→sku_master.sku_id
title	VARCHAR(250)	NO	G2	Must follow intent→cluster→attributes pattern. Validated by title engine.
description	TEXT	NO	Vector	Cosine similarity ≥ 0.72 vs cluster intent vector.
answer_block	VARCHAR(300)	COND	G4	Required for hero/support. 250-300 chars. Must contain primary intent keyword. NULL for harvest/kill.
best_for	JSON ARRAY	COND	G5	Required for hero/support. Min 2 entries. Array of strings.
not_for	JSON ARRAY	COND	G5	Required for hero/support. Min 1 entry. Array of strings.
expert_authority	TEXT	COND	G7	Required for hero/support. Must reference standard/cert. NULL for harvest/kill.
wikidata_uri	VARCHAR(100)	YES	-	Hero SKUs: recommended. Format: https://www.wikidata.org/entity/Q{ID}
material_id	VARCHAR(20)	YES	-	FK→material_wikidata.material_id. Used for JSON-LD material block.
vector_similarity	DECIMAL(6,4)	YES	-	Last computed cosine similarity. Updated on every save attempt.
updated_at	TIMESTAMP	NO	-	Auto-updated on content change.

sku_gate_status

One row per gate per SKU. Recalculated on every save/publish attempt. History in audit_log.

Field	Type	Null	Index	Notes
sku_id	VARCHAR(50)	NO	PK(comp),FK	FK→sku_master
gate_code	ENUM	NO	PK(comp)	Values: G1,G2,G3,G4,G5,G6,G6_1,G7,VECTOR. Composite PK with sku_id.
status	ENUM	NO	IDX	Values: pass, fail, not_applicable. Not_applicable for suspended gates (harvest mode).
error_code	VARCHAR(40)	YES	-	e.g. CIE_G4_CHAR_LIMIT. NULL if pass.
error_message	VARCHAR(500)	YES	-	User-facing message. NULL if pass.
checked_at	TIMESTAMP	NO	-	When this gate was last evaluated.

channel_readiness

One row per SKU per channel. Computed by scheduled job (daily).

Field	Type	Null	Index	Notes
sku_id	VARCHAR(50)	NO	PK(comp),FK	FK→sku_master
channel	ENUM	NO	PK(comp)	Values: google_sge, amazon, ai_assistants, own_website
score	SMALLINT	NO	IDX	0-100. Computed from component weights.
component_scores	JSON	NO	-	Breakdown: {cluster_id:10, intents:15, answer_block:20, ...}
computed_at	TIMESTAMP	NO	-	Last computation time.

ai_audit_runs + ai_audit_results + ai_golden_queries

Three tables for the weekly AI citation audit system.

```
-- ai_audit_runs: One row per weekly audit execution
CREATE TABLE ai_audit_runs (
  run_id        UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  category      ENUM('cables','lampshades','bulbs','pendants','floor_lamps') NOT NULL,
  run_date      DATE NOT NULL,
  status        ENUM('running','completed','failed') NOT NULL DEFAULT 'running',
  total_questions SMALLINT NOT NULL,
  aggregate_citation_rate DECIMAL(5,4),  -- e.g. 0.7500 = 75%
  pass_fail     ENUM('pass','fail','pending') DEFAULT 'pending',
  created_at    TIMESTAMP NOT NULL DEFAULT NOW()
);

-- ai_audit_results: One row per question per engine per run
```

```
CREATE TABLE ai_audit_results (
  result_id     UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  run_id        UUID NOT NULL REFERENCES ai_audit_runs(run_id),
  question_id   VARCHAR(20) NOT NULL REFERENCES ai_golden_queries(question_id),
  engine        ENUM('chatgpt','gemini','perplexity','google_sge') NOT NULL,
  score         SMALLINT NOT NULL CHECK (score BETWEEN 0 AND 3),
  response_snippet TEXT,  -- First 500 chars of AI response
  cited_sku_id  VARCHAR(50) REFERENCES sku_master(sku_id),
  created_at    TIMESTAMP NOT NULL DEFAULT NOW(),
  UNIQUE(run_id, question_id, engine)
);

-- ai_golden_queries: The locked 20-question sets
CREATE TABLE ai_golden_queries (
  question_id   VARCHAR(20) PRIMARY KEY,  -- e.g. CAB-Q01
  category      ENUM('cables','lampshades','bulbs','pendants','floor_lamps') NOT NULL,
  question_text VARCHAR(500) NOT NULL,
  intent_type   SMALLINT NOT NULL REFERENCES intent_taxonomy(intent_id),
  query_family  ENUM('primary','secondary','other') NOT NULL,
  target_tier   ENUM('hero','support') NOT NULL,
  target_skus   JSON NOT NULL,  -- Array of sku_ids
  success_criteria VARCHAR(300) NOT NULL,
  locked_until  DATE NOT NULL,  -- Cannot change before this date
  is_active     BOOLEAN NOT NULL DEFAULT true
);
```

sku_tier_history

Immutable. One row per tier change. Enables trend analysis and audit trail.

```
CREATE TABLE sku_tier_history (
  id            UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  sku_id        VARCHAR(50) NOT NULL REFERENCES sku_master(sku_id),
  old_tier      ENUM('hero','support','harvest','kill') NOT NULL,
  new_tier      ENUM('hero','support','harvest','kill') NOT NULL,
  reason        ENUM('erp_sync','manual_override','auto_promote','quarterly_review') NOT
NULL,
  approved_by   VARCHAR(100),  -- Required for manual_override
  second_approver VARCHAR(100),  -- Required for manual_override (dual sign-off)
  changed_at    TIMESTAMP NOT NULL DEFAULT NOW(),
  CHECK (old_tier != new_tier)
);
```

material_wikidata

Reference table. Maintained by SEO Governor. Used for JSON-LD generation.

```
CREATE TABLE material_wikidata (
  material_id   VARCHAR(20) PRIMARY KEY,  -- e.g. MAT-BORO-GLASS
  material_name VARCHAR(100) NOT NULL,  -- e.g. Borosilicate Glass
  wikidata_qid  VARCHAR(20) NOT NULL,  -- e.g. Q190117
  wikidata_uri  VARCHAR(100) NOT NULL,  -- https://www.wikidata.org/entity/Q190117
  ai_signal     VARCHAR(300) NOT NULL,  -- Why this matters for AI: 'Signals heat resistance'
  is_active     BOOLEAN NOT NULL DEFAULT true
);

-- Seed data:
INSERT INTO material_wikidata VALUES
('MAT-BORO-GLASS', 'Borosilicate Glass', 'Q190117',
 'https://www.wikidata.org/entity/Q190117',
 'Signals heat resistance and durability', true),
('MAT-OPAL-GLASS', 'Opal Glass', 'Q223425',
 'https://www.wikidata.org/entity/Q223425',
 'Signals light diffusion for soft light queries', true),
('MAT-COTTON', 'Cotton Fabric', 'Q11457',
 'https://www.wikidata.org/entity/Q11457',
 'Signals natural material for eco-conscious queries', true),
('MAT-BRASS', 'Brass', 'Q39782',
 'https://www.wikidata.org/entity/Q39782',
 'Signals premium and period-style', true),
('MAT-POLYCARB', 'Polycarbonate', 'Q146439',
 'https://www.wikidata.org/entity/Q146439',
 'Signals impact resistance and child safety', true);
```

audit_log (IMMUTABLE)

Append-only. No UPDATE or DELETE permitted. This table is the legal and compliance backbone of the enforcement system. See Document 7 for full spec.

```
CREATE TABLE audit_log (
  log_id        UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  entity_type   VARCHAR(30) NOT NULL,  -- sku, cluster, tier, content, gate, audit
  entity_id     VARCHAR(50) NOT NULL,  -- The ID of what changed
  action        ENUM('create','update','delete','publish','validate','tier_change',
                     'gate_pass','gate_fail','audit_run','brief_generated',
                     'escalation','login','permission_change') NOT NULL,
  field_name    VARCHAR(50),  -- Which field changed (NULL for non-field actions)
  old_value     TEXT,
  new_value     TEXT,
  actor_id      VARCHAR(100) NOT NULL,  -- User ID or 'SYSTEM'
  actor_role    VARCHAR(30) NOT NULL,  -- content_editor, seo_governor, admin, system, etc.
  ip_address    VARCHAR(45),
  user_agent    VARCHAR(300),
  timestamp     TIMESTAMP NOT NULL DEFAULT NOW(),
  -- NO UPDATE OR DELETE TRIGGERS ALLOWED ON THIS TABLE
  -- Enforced at DB level: REVOKE UPDATE, DELETE ON audit_log FROM ALL
);

CREATE INDEX idx_audit_entity ON audit_log(entity_type, entity_id);
CREATE INDEX idx_audit_actor ON audit_log(actor_id);
CREATE INDEX idx_audit_time ON audit_log(timestamp);
CREATE INDEX idx_audit_action ON audit_log(action);
```

Document 2: OpenAPI Specification

Full OpenAPI 3.0 YAML. Import directly into Swagger UI or Postman. Below is the complete spec — also provided as a separate .yaml file for direct import.

```yaml
openapi: '3.0.3'
info:
  title: CIE v2.3.1 - Catalog Intelligence Engine
  version: 2.3.1
  description: >
    Pre-publish enforcement gates, semantic validation, AI audit,
    and tier management for 120+ staff e-commerce operations.

servers:
  - url: https://cie.internal.example.com/api/v1
    description: Production
  - url: https://cie-staging.internal.example.com/api/v1
    description: Staging

security:
  - BearerAuth: []

paths:
  /sku/{sku_id}/validate:
    post:
      summary: Pre-publish validation (G1-G7 + Vector)
      description: >
        Called on every Save/Publish. Returns pass/fail per gate.
        Must respond within 500ms.
      tags: [Enforcement]
      parameters:
        - name: sku_id
          in: path
          required: true
          schema: { type: string, example: SKU-CABLE-001 }
      requestBody:
        required: true
        content:
          application/json:
            schema: { $ref: '#/components/schemas/SkuValidateRequest' }
      responses:
        '200':
          description: All gates pass — publish allowed
          content:
            application/json:
              schema: { $ref: '#/components/schemas/ValidationResponse' }
        '400':
          description: One or more gates failed — publish blocked
          content:
            application/json:
              schema: { $ref: '#/components/schemas/ValidationResponse' }

  /sku/{sku_id}/publish:
    post:
      summary: Publish SKU to channels
      description: >
        Only succeeds if all gates pass. Internally calls /validate first.
        Creates audit_log entry. Updates status to 'published'.
      tags: [Enforcement]
      parameters:
        - name: sku_id
          in: path
          required: true
          schema: { type: string }
      responses:
        '200': { description: Published successfully }
        '400': { description: Gate validation failed }
        '403': { description: Insufficient permissions }

  /sku/{sku_id}/readiness:
    get:
      summary: Get per-channel readiness scores
      tags: [Readiness]
      parameters:
        - name: sku_id
          in: path
          required: true
          schema: { type: string }
      responses:
        '200':
          content:
            application/json:
              schema: { $ref: '#/components/schemas/ReadinessResponse' }

  /sku/{sku_id}/embed:
    post:
      summary: Generate embedding for description text
      tags: [Semantic]
      requestBody:
        content:
          application/json:
            schema:
              type: object
              properties:
                text: { type: string, maxLength: 5000 }
              required: [text]
      responses:
        '200':
          content:
            application/json:
              schema:
                type: object
                properties:
                  vector: { type: array, items: { type: number } }
                  dimensions: { type: integer, example: 1536 }

  /sku/{sku_id}/similarity:
    post:
      summary: Check cosine similarity vs cluster intent
      tags: [Semantic]
      requestBody:
        content:
          application/json:
            schema:
              type: object
              properties:
                description: { type: string }
                cluster_id: { type: string }
              required: [description, cluster_id]
      responses:
        '200':
          content:
            application/json:
              schema: { $ref: '#/components/schemas/SimilarityResponse' }

  /taxonomy/intents:
    get:
      summary: Get intent taxonomy filtered by tier
      tags: [Taxonomy]
      parameters:
        - name: tier
          in: query
          schema:
            type: string
            enum: [hero, support, harvest, kill]
      responses:
        '200':
          content:
            application/json:
              schema:
                type: object
                properties:
                  intents:
                    type: array
                    items: { $ref: '#/components/schemas/Intent' }

  /clusters:
    get:
      summary: Get all active clusters
      tags: [Taxonomy]
      parameters:
        - name: category
          in: query
          schema: { type: string }
      responses:
        '200':
          content:
            application/json:
              schema:
                type: object
                properties:
                  clusters:
                    type: array
                    items: { $ref: '#/components/schemas/Cluster' }

  /audit/run:
    post:
      summary: Trigger AI citation audit for a category
      tags: [AI Audit]
      requestBody:
        content:
          application/json:
            schema:
              type: object
              properties:
                category:
                  type: string
                  enum: [cables, lampshades, bulbs, pendants, floor_lamps]
              required: [category]
      responses:
        '202': { description: Audit started, run_id returned }

  /audit/results/{category}:
    get:
      summary: Get latest audit results + decay status
      tags: [AI Audit]
      parameters:
        - name: category
          in: path
          required: true
          schema: { type: string }
        - name: run_id
          in: query
          schema: { type: string, format: uuid }
      responses:
        '200':
          content:
            application/json:
              schema: { $ref: '#/components/schemas/AuditResults' }

  /brief/generate:
    post:
      summary: Auto-generate content refresh brief (decay trigger)
      tags: [AI Audit]
      requestBody:
        content:
          application/json:
            schema:
              type: object
              properties:
                sku_id: { type: string }
                failing_questions: { type: array, items: { type: string } }
              required: [sku_id, failing_questions]
      responses:
        '201': { description: Brief created and queued }

  /erp/sync:
    post:
      summary: Receive ERP data push + recompute tiers
      tags: [ERP Integration]
      requestBody:
        content:
          application/json:
            schema: { $ref: '#/components/schemas/ErpSyncPayload' }
      responses:
        '200':
          content:
            application/json:
              schema:
                type: object
                properties:
                  skus_processed: { type: integer }
                  tier_changes: { type: integer }
                  errors: { type: array, items: { type: string } }

components:
  securitySchemes:
    BearerAuth:
      type: http
      scheme: bearer

  schemas:
    SkuValidateRequest:
      type: object
      required: [cluster_id, tier, primary_intent, title, description, action]
      properties:
        cluster_id: { type: string, example: CLU-CABLE-PENDANT-E27 }
        tier: { type: string, enum: [hero, support, harvest, kill] }
        primary_intent:
          type: string
          enum: [problem_solving, comparison, compatibility, specification,
                 installation, troubleshooting, inspiration, regulatory, replacement]
        secondary_intents:
          type: array
          items: { type: string }
          maxItems: 3
        title: { type: string, maxLength: 250 }
        description: { type: string }
        answer_block: { type: string, minLength: 250, maxLength: 300 }
        best_for: { type: array, items: { type: string }, minItems: 2 }
        not_for: { type: array, items: { type: string }, minItems: 1 }
        expert_authority: { type: string }
        action: { type: string, enum: [save, publish] }

    ValidationResponse:
      type: object
      properties:
        status: { type: string, enum: [pass, fail] }
        gates:
          type: object
          additionalProperties:
            type: object
            properties:
              status: { type: string, enum: [pass, fail, not_applicable] }
              error_code: { type: string, nullable: true }
              detail: { type: string }
              user_message: { type: string, nullable: true }
        vector_check:
          type: object
          properties:
            cosine_similarity: { type: number }
            threshold: { type: number, example: 0.72 }
            status: { type: string, enum: [pass, fail] }
        publish_allowed: { type: boolean }

    SimilarityResponse:
      type: object
      properties:
        cosine_similarity: { type: number, example: 0.87 }
        threshold: { type: number, example: 0.72 }
        status: { type: string, enum: [pass, fail] }
        message: { type: string, nullable: true }

    ReadinessResponse:
      type: object
      properties:
        sku_id: { type: string }
        channels:
          type: array
          items:
            type: object
            properties:
              channel: { type: string }
              score: { type: integer, minimum: 0, maximum: 100 }
              components: { type: object }

    ErpSyncPayload:
      type: object
      required: [sync_date, skus]
      properties:
        sync_date: { type: string, format: date-time }
        skus:
          type: array
          items:
            type: object
            required: [sku_id, contribution_margin_pct, cppc, velocity_90d, return_rate_pct]
            properties:
              sku_id: { type: string }
              contribution_margin_pct: { type: number }
              cppc: { type: number }
              velocity_90d: { type: integer }
              return_rate_pct: { type: number }
```

Document 3: RBAC & Permissions Matrix

Role-Based Access Control. This is what prevents 'just this once' overrides. No superuser bypass exists without dual sign-off from Finance Director + Commercial Director, and every such override is logged immutably.

3.1 Role Definitions

Role	Typical Staff	Purpose
content_editor	Content team (50+)	Creates/edits titles, descriptions, answer blocks, best-for/not-for within assigned SKUs.
product_specialist	Product experts (10+)	Adds expert authority blocks, safety certs, compliance data.
seo_governor	SEO lead (1-2)	Manages cluster master list, intent taxonomy, approves cluster assignments.
channel_manager	Channel leads (4-6)	Manages channel-specific content, feed optimisation, readiness scoring review.
ai_ops	AI operations (2-3)	Runs AI audits, manages golden queries, reviews citation decay alerts.
portfolio_holder	Category managers (8-12)	Reviews tier assignments, approves SKU status changes, owns category P&L.
finance	Finance team (2-3)	Provides ERP data, validates tier calculations, co-approves manual overrides.
admin	System admin (1-2)	System configuration, user management, audit log review. Cannot edit SKU content.
system	Automated processes	ERP sync, tier recomputation, decay triggers, readiness scoring. No human actor.

3.2 Permission Matrix

Action	Editor	Prod Spec	SEO Gov	Ch Mgr	AI Ops	PH	Finance	Admin	System
Create/edit content fields	**YES**	YES*	NO	YES*	NO	NO	NO	NO	NO
Edit expert authority	NO	**YES**	NO	NO	NO	NO	NO	NO	NO
Assign/change cluster_id	NO	NO	**YES**	NO	NO	NO	NO	NO	NO
Modify intent taxonomy	NO	NO	REVIEW*	NO	NO	NO	NO	NO	NO
Change SKU tier (manual)	NO	NO	NO	NO	NO	**DUAL****	**DUAL****	NO	AUTO
Publish SKU	**YES**	NO	YES	YES	NO	YES	NO	NO	NO

Run AI audit	NO	NO	NO	NO	**YES**	NO	NO	YES	YES
Manage golden queries	NO	NO	YES	NO	**YES**	NO	NO	NO	NO
View audit logs	OWN	OWN	ALL	OWN	ALL	CAT	ALL	**ALL**	N/A
Manage users/roles	NO	NO	NO	NO	NO	NO	NO	**YES**	NO
ERP sync trigger	NO	NO	NO	NO	NO	NO	YES	YES	**YES**

***** Product Specialist and Channel Manager can edit content ONLY within their assigned categories.

*** REVIEW:** SEO Governor can propose taxonomy changes; requires quarterly review with Commercial Director to activate.

**** DUAL:** Manual tier override requires BOTH Portfolio Holder AND Finance sign-off. Both must approve in the system before the change takes effect. Creates audit_log entry with both approver IDs.

CRITICAL RBAC RULES

1. No 'superuser' role exists. Admin can manage users but CANNOT edit SKU content or bypass gates.

2. Content editors CANNOT change cluster_id, tier, or intent taxonomy.

3. Kill-tier SKUs: ALL edit permissions are revoked system-wide. Only tier change (DUAL approval) can unlock.

4. Every permission check is logged in audit_log with actor_id, role, action, and timestamp.

5. Failed permission attempts (403) are logged and monitored. >5 in 24hrs triggers admin alert.

Document 4: Test Fixtures & Golden Dataset

30 SKU fixtures with expected gate outcomes. Every gate, every tier, every edge case. Dev team must write automated tests against these. 100% pass rate required before go-live.

4.1 Fixture Summary

#	SKU ID	Tier	Category	Expected	Tests
1	SKU-CABLE-001	hero	cables	**ALL PASS**	Full CIE stack. All 7 gates + vector pass.
2	SKU-SHADE-001	hero	lampshades	**ALL PASS**	Hero with Wikidata + Expert Authority.
3	SKU-BULB-001	support	bulbs	**ALL PASS**	Support tier. 2 secondary intents max.
4	SKU-PEND-001	harvest	pendants	**ALL PASS**	Harvest maintenance mode. Reduced gates.
5	SKU-CABLE-002	hero	cables	**G1 FAIL**	Invalid cluster_id: 'CUSTOM-CABLE-XYZ' (not in master list).
6	SKU-SHADE-002	hero	lampshades	**G2 FAIL**	Primary intent = 'mood_lighting' (not in taxonomy).
7	SKU-BULB-002	support	bulbs	**G3 FAIL**	Secondary intent same as primary (duplicate).
8	SKU-PEND-002	hero	pendants	**G4 FAIL**	Answer block = 312 chars (over 300 limit).
9	SKU-CABLE-003	hero	cables	**G4 FAIL**	Answer block missing primary intent keyword.
10	SKU-SHADE-003	hero	lampshades	**G5 FAIL**	Only 1 best_for entry (min 2 required).
11	SKU-BULB-003	support	bulbs	**G5 FAIL**	Zero not_for entries (min 1 required).
12	SKU-PEND-003	-	pendants	**G6 FAIL**	No tier tag assigned.
13	SKU-CABLE-004	harvest	cables	**G6.1 FAIL**	Harvest SKU with troubleshooting intent (blocked for harvest).
14	SKU-SHADE-004	kill	lampshades	**G6.1 FAIL**	Kill SKU: any edit attempt blocked.
15	SKU-BULB-004	hero	bulbs	**G7 FAIL**	Hero SKU with empty expert_authority.
16	SKU-PEND-004	hero	pendants	**VEC FAIL**	Description cosine similarity = 0.58 (below 0.72).
17	SKU-CABLE-005	support	cables	**G3 FAIL**	4 secondary intents (max 2 for support).

18	SKU-SHADE-005	harvest	lampshades	**PASS***	Harvest: G4,G5,G7 = not_applicable. Only G1,G2,G6,G6.1 checked.
19	SKU-BULB-005	hero	bulbs	**MULTI FAIL**	G4 + G5 + G7 all fail simultaneously. Tests multiple error return.
20	SKU-PEND-005	hero	pendants	**ALL PASS**	Edge: answer_block exactly 250 chars (minimum).
21	SKU-CABLE-006	hero	cables	**ALL PASS**	Edge: answer_block exactly 300 chars (maximum).
22	SKU-SHADE-006	hero	lampshades	**G4 FAIL**	Edge: answer_block 249 chars (1 below minimum).
23	SKU-BULB-006	hero	bulbs	**G4 FAIL**	Edge: answer_block 301 chars (1 above maximum).
24-30	SKU-*-007 to 010	mixed	mixed	**MIXED**	Permission tests: wrong role attempts cluster change, tier change without dual approval, kill edit attempt. All must return 403 + audit log entry.

TEST AUTOMATION REQUIREMENTS

1. All 30 fixtures must have automated tests (PHPUnit for PHP gates, pytest for Python services).

2. Tests run on every CI/CD build. Any failure blocks deployment.

3. Fixture data is loaded into test DB before each run and torn down after.

4. Vector similarity tests use pre-computed embeddings (no live API calls in CI).

5. Permission tests use mock auth tokens for each role.

6. 100% pass rate required before go-live. No exceptions.

Document 5: Sequence Diagrams

Four core workflows shown as text-based sequence diagrams. Dev team should convert to Mermaid/PlantUML for their wiki.

5.1 SKU Create/Edit Flow

```
User (CMS)          PHP CMS                Python Engine          Database
    │                    │                      │                     │
    │  Fill form fields  │                      │                     │
    │───────────────────→│                      │                     │
    │                    │                      │                     │
    │                    │  Check RBAC          │                     │
    │                    │─────────────────────→│                     │
    │                    │  (role vs action)    │                     │
    │                    │←─────────────────────│                     │
    │                    │                      │                     │
    │                    │  GET /taxonomy/intents│                    │
    │                    │  ?tier={sku_tier}    │                     │
    │                    │─────────────────────→│                     │
    │                    │  Return allowed intents                    │
    │                    │←─────────────────────│                     │
    │                    │                      │                     │
    │  [User fills fields visible for tier]     │                     │
    │                    │                      │                     │
    │  Click 'Save'      │                      │                     │
    │───────────────────→│                      │                     │
    │                    │  POST /sku/{id}/validate                   │
    │                    │─────────────────────→│                     │
    │                    │                      │  Compute embedding  │
    │                    │                      │  Check G1-G7+G6.1   │
    │                    │                      │  Check vector sim   │
    │                    │  Return gate results │                     │
    │                    │←─────────────────────│                     │
    │                    │                      │                     │
    │                    │  [If ALL pass]       │                     │
    │                    │  Save to DB          │                     │
    │                    │──────────────────────────────────────────→│
    │                    │  Write audit_log     │                     │
    │                    │──────────────────────────────────────────→│
    │  'Saved OK'        │                      │                     │
    │←───────────────────│                      │                     │
    │                    │                      │                     │
    │                    │  [If ANY fail]       │                     │
    │  Show gate errors  │                      │                     │
    │  (inline, per field)                      │                     │
    │←───────────────────│                      │                     │
```

5.2 Publish Attempt Flow

```
User (CMS)          PHP CMS                Python Engine          Database
    │                    │                      │                     │
    │  Click 'Publish'   │                      │                     │
    │───────────────────→│                      │                     │
    │                    │  Check RBAC          │                     │
    │                    │  (can user publish?) │                     │
    │                    │                      │                     │
    │                    │  POST /sku/{id}/validate                   │
    │                    │  action='publish'    │                     │
    │                    │─────────────────────→│                     │
    │                    │                      │  Run ALL gates      │
    │                    │                      │  (stricter than save)│
    │                    │  Gate results        │                     │
    │                    │←───────────────────────│                    │
    │                    │                       │                    │
    │                    │  [ALL PASS]            │                    │
    │                    │  UPDATE status='published'                  │
    │                    │──────────────────────────────────────────→│
    │                    │  Write audit_log(action='publish')          │
    │                    │──────────────────────────────────────────→│
    │                    │  Trigger channel feeds  │                    │
    │                    │───────────────────────→│                    │
    │                    │  (generate JSON-LD,     │                    │
    │                    │   push to Amazon/Google)│                    │
    │  'Published'        │                       │                    │
    │←───────────────────│                       │                    │
    │                    │                       │                    │
    │                    │  [ANY FAIL]            │                    │
    │  Show blocking errors │                     │                    │
    │  'Cannot publish: G4 Answer Block too long'│                     │
    │←───────────────────│                       │                    │
```

5.3 Weekly AI Audit Job

```
N8N Scheduler        Python Engine          AI APIs              Database
    │                    │                       │                    │
    │  CRON: Mon 06:00   │                       │                    │
    │──────────────────→│                       │                    │
    │                    │  Load golden queries    │                    │
    │                    │  for category           │                    │
    │                    │←──────────────────────────────────────────│
    │                    │                       │                    │
    │                    │  FOR each question:    │                    │
    │                    │    FOR each engine:    │                    │
    │                    │      Submit query       │                    │
    │                    │───────────────────────→│                    │
    │                    │      AI response        │                    │
    │                    │←───────────────────────│                    │
    │                    │      Score 0-3          │                    │
    │                    │      Store result       │                    │
    │                    │──────────────────────────────────────────→│
    │                    │                       │                    │
    │                    │  Compute aggregate     │                    │
    │                    │  citation rate          │                    │
    │                    │                       │                    │
    │                    │  Compare to prev 2 wks│                     │
    │                    │  Update decay_status    │                    │
    │                    │──────────────────────────────────────────→│
    │                    │                       │                    │
    │                    │  [If decay_status =     │                    │
    │                    │   'auto_brief']         │                    │
    │                    │  Generate refresh brief│                     │
    │                    │  Add to content queue   │                    │
    │                    │  Send alerts            │                    │
    │                    │                       │                    │
    │  Job complete       │                       │                    │
    │←──────────────────│                       │                    │
```

5.4 ERP Sync + Tier Recomputation

```
ERP System           Python Engine          Database             PHP CMS
    │                    │                       │                    │
    │  POST /erp/sync    │                       │                    │
    │  (monthly push)    │                       │                    │
    │──────────────────→│                       │                    │
```

```
     │                    │  Parse payload        │                    │
     │                    │  Update sku_master    │                    │
     │                    │  (margin, cppc, vel)  │                    │
     │                    │──────────────────────────────────────────→│
     │                    │                       │                    │
     │                    │  Compute percentiles  │                    │
     │                    │  FOR each SKU:        │                    │
     │                    │    Calculate score    │                    │
     │                    │    Determine new tier │                    │
     │                    │    IF tier changed:   │                    │
     │                    │      Update sku_master│                    │
     │                    │      Insert tier_history│                  │
     │                    │      Insert audit_log │                    │
     │                    │──────────────────────────────────────────→│
     │                    │                       │                    │
     │                    │  Check auto-promote   │                    │
     │                    │  (harvest → support   │                    │
     │                    │   if velocity +30%)   │                    │
     │                    │                       │                    │
     │                    │  Notify CMS of changes│                    │
     │                    │──────────────────────────────────────────→│
     │                    │                       │    Update UI tier  │
     │                    │                       │    restrictions    │
     │  Sync complete     │                       │                    │
     │←───────────────────│                       │                    │
```

Document 6: Title Construction Engine

Pseudocode for generating intent-first titles. The PHP CMS renders a title preview as the user fills in fields. The final title is validated at save/publish (G2 gate).

6.1 Title Formula

```
TITLE = [INTENT_PHRASE] + ' | ' + [PRODUCT_CLASS] + ' ' + [KEY_ATTRIBUTES]

Where:
  INTENT_PHRASE  = Derived from primary_intent + cluster_intent_statement
  PRODUCT_CLASS  = From cluster category mapping
  KEY_ATTRIBUTES = Physical specs (material, size, fitting, colour)

Max length: 250 characters (including separator)
First segment (before |): Must address the user's problem/intent
Second segment (after |): Physical identification for specification queries
```

6.2 Intent Phrase Templates by Product Class

Category	Primary Intent	Intent Phrase Template	Example Output
Cables	Compatibility	{Product} for {Target Fitting} — {Benefit}	Pendant Cable Set for Ceiling Lights — Safe Wiring Made Simple
Cables	Installation	Easy-Install {Product} for {Application}	Easy-Install Pendant Cable Kit for DIY Ceiling Lighting
Lampshades	Problem-Solving	{Benefit} for {Room/Context}	Warm Glare-Free Lighting for Living Rooms
Lampshades	Comparison	{Material A} vs {Material B} — {Decision Help}	Fabric vs Glass Shade — Which Gives Better Reading Light
Lampshades	Replacement	Replacement {Product} for {Existing Setup}	Replacement Drum Shade for Floor Lamps and Pendants
Bulbs	Compatibility	{Bulb Type} for {Fitting} — {Key Spec}	LED Bulb for E27 Pendant and Table Lamps — Warm Filament Glow
Bulbs	Specification	{Output} {Bulb Type} — {Equivalence}	470 Lumen LED Filament — 40W Incandescent Equivalent
Pendants	Inspiration	{Style} {Product} Ideas for {Space}	Modern Pendant Lighting Ideas for Kitchen Islands
Pendants	Regulatory	{Safety Standard} {Product} for {Environment}	IP44 Rated Pendant Light for Bathroom Zones

6.3 Title Generation Pseudocode

```
def generate_title(sku):
    cluster = get_cluster(sku.cluster_id)
    intent = get_intent(sku.primary_intent_id)
    template = get_title_template(cluster.category, intent.intent_key)

    # Phase 1: Generate intent phrase from template
    intent_phrase = template.render(
        product=cluster.product_class_label,
        target_fitting=sku.fitting_type,
        benefit=extract_primary_benefit(cluster.intent_statement),
        room_context=sku.primary_use_context,
        material=sku.material_name
    )

    # Phase 2: Build attribute string
    attrs = []
    if sku.material_name: attrs.append(sku.material_name)
    if sku.primary_dimension: attrs.append(sku.primary_dimension)
    if sku.fitting_type: attrs.append(sku.fitting_type)
    if sku.colour and sku.colour != 'N/A': attrs.append(sku.colour)
    attribute_str = ' '.join(attrs)

    # Phase 3: Assemble
    title = f'{intent_phrase} | {attribute_str}'

    # Phase 4: Validate
    if len(title) > 250:
        # Truncate attributes, never truncate intent phrase
        title = truncate_attributes(intent_phrase, attrs, max_len=250)

    if len(intent_phrase) < 20:
        raise ValidationError('Intent phrase too short. Must address a problem.')

    return title

# VALIDATION RULE (G2 gate):
# Title must contain '|' separator
# Text before '|' must NOT start with a colour, material, or dimension
# Text before '|' must contain at least one intent-related word
#   (from intent keyword dictionary)
```

Document 7: Immutable Audit Logging Specification

Every change to every entity is logged permanently. No UPDATE or DELETE is permitted on the audit_log table. This is the enforcement credibility backbone — without it, 'no override' is unverifiable.

7.1 What Gets Logged

Event	entity_type	action	What's Captured
SKU content edit	sku	update	field_name, old_value, new_value
SKU publish attempt	sku	publish	All gate statuses at time of publish
Gate pass	gate	gate_pass	gate_code, sku_id
Gate fail	gate	gate_fail	gate_code, sku_id, error_code, error_message
Tier change	tier	tier_change	old_tier, new_tier, reason, approver(s)
Cluster assignment	cluster	update	old_cluster_id, new_cluster_id
AI audit run	audit	audit_run	category, run_id, aggregate_score
Auto-brief generated	audit	brief_generated	sku_id, failing_questions, deadline
Escalation triggered	audit	escalation	sku_id, decay_weeks, escalated_to
Permission denied (403)	auth	permission_change	attempted_action, required_role, actual_role
User login	auth	login	actor_id, ip_address, user_agent
ERP sync	erp	create	skus_processed, tier_changes, errors

7.2 Database Enforcement

```
-- Prevent any modification to audit_log
REVOKE UPDATE, DELETE ON audit_log FROM PUBLIC;
REVOKE UPDATE, DELETE ON audit_log FROM app_user;
REVOKE UPDATE, DELETE ON audit_log FROM admin_user;

-- Only the app service account can INSERT
GRANT INSERT ON audit_log TO cie_service_account;

-- Trigger to block any UPDATE/DELETE attempt
CREATE OR REPLACE FUNCTION prevent_audit_modification()
RETURNS TRIGGER AS $$
```

```
BEGIN
  RAISE EXCEPTION 'audit_log is immutable. UPDATE/DELETE not permitted.';
  RETURN NULL;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER audit_log_immutable
  BEFORE UPDATE OR DELETE ON audit_log
  FOR EACH ROW EXECUTE FUNCTION prevent_audit_modification();
```

7.3 Retention & Compliance

Retention: Minimum 3 years. No automatic purge.

Backup: Daily incremental, weekly full. Stored separately from main DB backups.

Access: Read-only for admin, seo_governor, finance, ai_ops. Content editors see only their own entries.

Monitoring: >5 gate_fail events from same actor in 24hrs triggers admin alert. >10 permission denied events in 24hrs triggers security review.

Document 8: Channel Output Mapping

Defines exactly which CIE fields map to each sales channel, including format transformations, character limits, and where JSON-LD/structured data is injected.

8.1 Field Mapping by Channel

CIE Field	Own Website	Google Shopping	Amazon	AI Assistants
title	H1 tag + <title>	product_title (150 char max)	product_name (200 char max)	Via JSON-LD name
answer_block	FAQ schema + above fold	Not used directly	Not used directly	JSON-LD description
description	Product description body	description (5000 char)	product_description (2000)	Via crawl/schema
best_for	'Ideal For' section	Not in feed	Bullet points 1-2	JSON-LD additionalProperty
not_for	'Not Suitable For' section	Not in feed	Not in listing	JSON-LD additionalProperty
expert_authority	Trust badge + footer	Not in feed	'From the manufacturer'	JSON-LD additionalProperty
primary_intent	Internal: URL structure	custom_label_0	search_terms (backend)	Not exposed directly
cluster_id	Internal: breadcrumbs	product_type	browse_node	Not exposed directly
tier	Internal: display priority	custom_label_1	Not used	Not exposed
wikidata_uri	JSON-LD sameAs	Not in feed	Not used	JSON-LD material.sameAs
JSON-LD (full)	Injected in <head>	Via Merchant Center	Not applicable	Crawled from website

8.2 Amazon-Specific Transformations

```
# Amazon Backend Search Terms (250 char limit)
# Generated from CIE data, NOT manually entered

def generate_amazon_search_terms(sku):
    terms = []

    # Add intent-derived keywords
    intent = get_intent(sku.primary_intent_id)
    terms.extend(intent.amazon_keywords)   # Pre-mapped per intent

    # Add compatibility terms
    if sku.fitting_type:
        terms.append(sku.fitting_type)   # e.g. 'E27', 'B22'
    # Add use-case terms from best_for
    for bf in sku.best_for[:3]:
        terms.extend(extract_keywords(bf))

    # Deduplicate and truncate to 250 chars
    unique_terms = list(dict.fromkeys(terms))
    return ' '.join(unique_terms)[:250]

# Amazon Bullet Points (5 bullets, 500 chars each)
# Bullet 1-2: From best_for
# Bullet 3: From expert_authority (safety/compliance)
# Bullet 4: From primary intent answer (condensed)
# Bullet 5: Compatibility statement
```

8.3 Google Shopping Feed Mapping

```
# Google Merchant Center Feed Fields

feed_mapping = {
    'id': sku.sku_id,
    'title': truncate(sku.title, 150),  # Google max 150 chars
    'description': sku.description[:5000],
    'product_type': cluster_to_google_taxonomy(sku.cluster_id),
    'custom_label_0': sku.primary_intent,  # For campaign segmentation
    'custom_label_1': sku.tier,  # Hero/Support for bid strategy
    'custom_label_2': sku.cluster_id,  # For reporting
}

# Readiness check before feed inclusion:
# - Hero SKUs: must have readiness >= 85 for google_sge channel
# - Support SKUs: must have readiness >= 70
# - Harvest/Kill: excluded from Shopping feed entirely
```

8.4 Own Website JSON-LD Injection Point

Location: <head> section of product page, before closing </head> tag.

Template: Generated by PHP render_cie_jsonld() function (see v2.3.1 Section 11.1).

FAQ Schema: For Hero SKUs, inject additional FAQPage schema using answer_block content + top 3 golden queries for that SKU's cluster.

Breadcrumbs: Generated from cluster_id → category → product_class hierarchy.

Pre-Launch Checklist

Dev lead signs off each item. All must be YES before go-live.

#	Check	Status	Owner	Evidence
1	All 12 DB tables created with correct types, FKs, and constraints	[ ]	DB Dev	Migration scripts
2	All API endpoints respond per OpenAPI spec (Swagger test)	[ ]	Backend Dev	Postman collection
3	All 30 test fixtures pass (PHPUnit + pytest)	[ ]	QA	CI/CD output
4	RBAC enforced: content_editor cannot change cluster, tier, or taxonomy	[ ]	Backend Dev	Permission tests
5	Kill-tier SKUs: all edit attempts return 403	[ ]	Backend Dev	Fixture #14
6	audit_log immutable: UPDATE/DELETE attempts blocked at DB level	[ ]	DB Dev	DB trigger test
7	Vector validation service responds <500ms for single SKU	[ ]	Python Dev	Load test
8	CMS UI hides/shows fields correctly per tier (G6.1)	[ ]	Frontend Dev	UI test per tier
9	JSON-LD passes Google Rich Results Test for Hero SKUs	[ ]	SEO + Dev	Google tool output
10	Amazon feed generates correct search terms + bullets from CIE data	[ ]	Integration Dev	Sample feed review
11	N8N audit workflow runs successfully in staging	[ ]	AI Ops + Dev	N8N execution log
12	Decay trigger fires correctly at Week 3 (auto-brief generated)	[ ]	Python Dev	Test run output
13	ERP sync correctly recomputes tiers + creates tier_history entries	[ ]	Python Dev	Sync test output
14	Readiness dashboard displays correct scores for all 4 channels	[ ]	Frontend Dev	Dashboard review
15	Material Wikidata seed data loaded (5 materials)	[ ]	SEO Gov	DB query

KILL RULE
If items 1-6 are not complete by Day 7 of the sprint, STOP all other development. Enforcement gates and data integrity are the foundation. Nothing else works without them.
Do not proceed to content engine, AI audit, or channel feeds until the gates are provably blocking.

This is the complete build pack.

8 documents. Every table, every endpoint, every permission, every test case, every channel mapping. Your dev team should be able to start coding on Day 1 without asking a single clarification question about data structures, API contracts, or business rules.

Combined with CIE v2.3.1 Enforcement + Dev Spec, this is the total package.