

Contents

RAWS v1.1 Engineering Documentation	1
Technical Reference for Engineers and Developers	1
1. System Overview	1
2. Weight Profile System	2
2.1 Profile Categories	2
2.2 Complete Profile Matrix	2
2.3 Domain Profile Details	2
3. Eight-Dimension Scoring	4
3.1 Dimension Calculations	4
3.2 Composite Score	4
4. Selection Algorithm	4
4.1 Four Phases	4
4.2 Weight Resolution Order	5
5. Domain Detection	6
5.1 Keyword Detection	6
5.2 Task Type Mapping	6
6. TypeScript Implementation	6
6.1 Profile Types	6
6.2 Domain to Profile Mapping	7
7. Database Schema	7
7.1 Weight Profiles Table	7
7.2 Domain Config Table	8
8. API Examples	8
8.1 Domain-Specific Selection	8
8.2 Auto-Detection	9
9. Testing	9
9.1 Profile Validation	9
10. Performance	10
10.1 Latency Budget	10

RAWS v1.1 Engineering Documentation

Technical Reference for Engineers and Developers

Document Version: 1.1.0

RADIANT Platform Version: v4.19.0

Last Updated: January 2026

1. System Overview

RAWS (RADIANT AI Weighted Selection) is the real-time model orchestration system that selects optimal AI models using:

- **8-Dimension Scoring:** Quality, Cost, Latency, Capability, Reliability, Compliance, Availability, Learning
 - **13 Weight Profiles:** 4 Optimization + 6 Domain + 3 SOFAI
 - **7 Domains:** Healthcare, Financial, Legal, Scientific, Creative, Engineering, General
 - **106+ Models:** 50 external APIs + 56 self-hosted
-

2. Weight Profile System

2.1 Profile Categories

Category	Count	Purpose
Optimization	4	General optimization strategies
Domain	6	Domain-specific requirements
SOFIAI	3	Cognitive complexity routing

2.2 Complete Profile Matrix

Profile	Q	C	L	K	R	P	A	E	Focus
BALANCED	0.25	0.20	0.15	0.15	0.10	0.05	0.05	0.05	Default
QUALITY_FIRST	0.10	0.10	0.15	0.15	0.10	0.05	0.05	0.05	Max accuracy
COST_OPTIMIZED	0.35	0.15	0.10	0.05	0.05	0.05	0.05	0.05	Min cost
LATENCY_CRITICAL	0.35	0.35	0.15	0.10	0.05	0.05	0.05	0.05	Fastest
HEALTHCARE	0.05	0.10	0.15	0.10	0.20	0.05	0.05	0.05	Quality+Compliance
FINANCIAL	0.30	0.10	0.10	0.15	0.10	0.15	0.05	0.05	Accuracy+Audit
LEGAL	0.35	0.05	0.05	0.20	0.10	0.15	0.05	0.05	Citations
SCIENTIFIC	0.35	0.10	0.10	0.20	0.08	0.05	0.05	0.07	Research
CREATIVE	0.20	0.25	0.20	0.15	0.05	0.00	0.05	0.10	Iteration
ENGINEERING	0.15	0.15	0.20	0.10	0.10	0.00	0.05	0.05	Code
SYSTEM_1	0.15	0.30	0.30	0.10	0.05	0.00	0.05	0.05	Fast+Cheap
SYSTEM_2	0.35	0.10	0.10	0.15	0.10	0.10	0.05	0.05	Reasoning
SYSTEM_20	0.05	0.05	0.05	0.20	0.10	0.10	0.05	0.05	Max reasoning

2.3 Domain Profile Details

HEALTHCARE

- **Weights:** Q=0.30, C=0.05, L=0.10, K=0.15, R=0.10, P=0.20, A=0.05, E=0.05
- **Constraints:** minQuality=80, compliance=[HIPAA], systemType=SYSTEM_2
- **Truth Engine:** Required (ECD threshold: 0.05)
- **Regulatory Requirements:**
 - **HIPAA:** Mandatory for Protected Health Information (PHI). Requires encryption, access controls, audit trails, BAAs.

- **FDA 21 CFR Part 11:** Required for clinical trials, drug development, medical device decisions.
- **HITECH Act:** Extends HIPAA for electronic health records.
- **Use Cases:** Medical diagnosis, patient data analysis, clinical documentation

FINANCIAL

- **Weights:** Q=0.30, C=0.10, L=0.10, K=0.15, R=0.10, P=0.15, A=0.05, E=0.05
- **Constraints:** minQuality=75, compliance=[SOC2], systemType=SYSTEM_2
- **Truth Engine:** Required (ECD threshold: 0.05)
- **Regulatory Requirements:**
 - **SOC 2 Type II:** Required for security controls, availability, processing integrity, confidentiality.
 - **PCI-DSS:** Required if processing payment card data.
 - **GDPR:** Required for EU resident financial data.
 - **SEC/FINRA:** Investment advice faces regulatory scrutiny.
 - **SOX:** Audit trail requirements for public companies.
- **Use Cases:** Investment analysis, accounting, financial reporting

LEGAL

- **Weights:** Q=0.35, C=0.05, L=0.05, K=0.20, R=0.10, P=0.15, A=0.05, E=0.05
- **Constraints:** minQuality=80, compliance=[SOC2], systemType=SYSTEM_2
- **Truth Engine:** Required, source citation required (ECD threshold: 0.05)
- **Regulatory Requirements:**
 - **SOC 2 Type II:** Required for attorney-client privilege protection, confidential documents.
 - **ABA Model Rules:** AI legal research must meet professional responsibility standards.
 - **GDPR:** Required for EU data subjects in legal matters.
 - **State Bar Requirements:** Many jurisdictions require AI use disclosure.
- **Use Cases:** Contract analysis, legal research, compliance documentation

SCIENTIFIC

- **Weights:** Q=0.35, C=0.10, L=0.10, K=0.20, R=0.08, P=0.05, A=0.05, E=0.07
- **Constraints:** minQuality=70, source citation required
- **Truth Engine:** Optional (ECD threshold: 0.08)
- **Regulatory Requirements:**
 - **FDA 21 CFR Part 11:** Required for pharmaceutical/drug research.
 - **GLP (Good Laboratory Practice):** For studies submitted to regulatory agencies.
 - **IRB Approval:** Human subjects research may require institutional review.
 - **NIH Data Management:** Data integrity requirements for federally funded research.
- **Use Cases:** Research analysis, data interpretation, peer review assistance

CREATIVE

- **Weights:** Q=0.20, C=0.25, L=0.20, K=0.15, R=0.05, P=0.00, A=0.05, E=0.10
- **Constraints:** None (most flexible)
- **Truth Engine:** Not required (ECD threshold: 0.20)

- **Regulatory Requirements:**
 - **None required:** Creative content not subject to regulatory compliance.
 - **FTC Guidelines:** Disclosures may be required for AI-generated advertising.
- **Use Cases:** Content writing, storytelling, brainstorming, marketing copy

ENGINEERING

- **Weights:** Q=0.30, C=0.15, L=0.15, K=0.20, R=0.10, P=0.00, A=0.05, E=0.05
 - **Constraints:** minQuality=70, preferredCapabilities=[function_calling, tool_use]
 - **Truth Engine:** Optional (ECD threshold: 0.10)
 - **Regulatory Requirements:**
 - **SOC 2 Type II:** Required if AI-generated code processes sensitive data.
 - **ISO 27001:** Information security management for enterprise software.
 - **NIST Cybersecurity Framework:** Recommended for security-sensitive applications.
 - **FDA 21 CFR Part 11:** Required for medical device software.
 - **IEC 62443:** Required for industrial control systems.
 - **Use Cases:** Code generation, code review, debugging, architecture design
-

3. Eight-Dimension Scoring

3.1 Dimension Calculations

Dimension	Formula	Range
Quality (Q)	Weighted benchmark average	0-100
Cost (C)	Inverted normalized price	0-100
Latency (L)	TTFT threshold mapping	0-100
Capability (K)	matched / required × 100	0-100
Reliability (R)	Uptime + error rate composite	0-100
Compliance (P)	Framework count × 15	0-100
Availability (A)	Thermal state mapping	0-100
Learning (E)	Historical performance	0-100

3.2 Composite Score

$$\text{CompositeScore} = Q \times Wq + C \times Wc + L \times Wl + K \times Wk + R \times Wr + P \times Wp + A \times Wa + E \times We$$

Where $\Sigma(\text{weights}) = 1.0$

4. Selection Algorithm

4.1 Four Phases

PHASE 1: FILTER (5ms)

- Status filter (active only)
- Capability filter
- Compliance filter

Tier filter
 System type filter
 Thermal filter

PHASE 2: SCORE (25ms)

- Quality scorer
- Cost scorer
- Latency scorer
- Capability scorer
- Reliability scorer
- Compliance scorer
- Availability scorer
- Learning scorer (parallel)

PHASE 3: RANK (15ms)

- Apply weights from profile
- Calculate composite scores
- Apply neural adjustments
- Sort by adjusted score

PHASE 4: SELECT (3ms)

- Select winner
- Select fallbacks (3)
- Generate reason
- Build response

4.2 Weight Resolution Order

```

async resolveWeights(request, systemType, domain): Promise<ScoringWeights> {
  // 1. Explicit profile ID
  if (request.weightProfileId) {
    return getProfile(request.weightProfileId).weights;
  }

  // 2. Optimization preference
  if (request.optimizeFor) {
    return getOptimizationProfile(request.optimizeFor).weights;
  }

  // 3. Domain-specific (includes SCIENTIFIC, CREATIVE, ENGINEERING)
  if (domain !== 'general') {
    return getDomainProfile(domain).weights;
  }

  // 4. SOFAI system type
  return getSystemProfile(systemType).weights;
}

```

5. Domain Detection

5.1 Keyword Detection

```
const DOMAIN_KEYWORDS = {
  healthcare: ['medical', 'diagnosis', 'patient', 'clinical', ...],
  financial: ['investment', 'stock', 'trading', 'accounting', ...],
  legal: ['contract', 'lawsuit', 'attorney', 'litigation', ...],
  scientific: ['research', 'experiment', 'hypothesis', 'study', ...],
  creative: ['write', 'story', 'creative', 'brainstorm', ...],
  engineering: ['code', 'programming', 'debug', 'api', ...],
};
```

5.2 Task Type Mapping

```
const TASK_TYPE_MAP = {
  'medical_qa': 'healthcare',
  'clinical_documentation': 'healthcare',
  'investment_analysis': 'financial',
  'contract_analysis': 'legal',
  'research_analysis': 'scientific',
  'content_writing': 'creative',
  'code_generation': 'engineering',
  'code_review': 'engineering',
  'debugging': 'engineering',
};
```

6. TypeScript Implementation

6.1 Profile Types

```
export type OptimizationProfile =
  | 'BALANCED'
  | 'QUALITY_FIRST'
  | 'COST_OPTIMIZED'
  | 'LATENCY_CRITICAL';

export type DomainProfile =
  | 'HEALTHCARE'
  | 'FINANCIAL'
  | 'LEGAL'
  | 'SCIENTIFIC'
  | 'CREATIVE'
  | 'ENGINEERING';

export type SOFAIProfile =
```

```

| 'SYSTEM_1'
| 'SYSTEM_2'
| 'SYSTEM_2_5';

export type WeightProfileId =
| OptimizationProfile
| DomainProfile
| SOFAIPProfile;

export type Domain =
| 'healthcare'
| 'financial'
| 'legal'
| 'scientific'
| 'creative'
| 'engineering'
| 'general';

```

6.2 Domain to Profile Mapping

```

export const DOMAIN_PROFILE_MAP: Record<Domain, WeightProfileId> = {
  healthcare: 'HEALTHCARE',
  financial: 'FINANCIAL',
  legal: 'LEGAL',
  scientific: 'SCIENTIFIC',
  creative: 'CREATIVE',
  engineering: 'ENGINEERING',
  general: 'BALANCED',
};

```

7. Database Schema

7.1 Weight Profiles Table

```

CREATE TABLE raws_weight_profiles (
  id VARCHAR(50) PRIMARY KEY,
  display_name VARCHAR(255) NOT NULL,
  description TEXT,
  category VARCHAR(20) NOT NULL, -- 'optimization', 'domain', 'sofai'

  -- Eight dimension weights
  weight_quality NUMERIC(4, 3) NOT NULL,
  weight_cost NUMERIC(4, 3) NOT NULL,
  weight_latency NUMERIC(4, 3) NOT NULL,
  weight_capability NUMERIC(4, 3) NOT NULL,
  weight_reliability NUMERIC(4, 3) NOT NULL,
  weight_compliance NUMERIC(4, 3) NOT NULL,

```

```

    weight_availability NUMERIC(4, 3) NOT NULL,
    weight_learning NUMERIC(4, 3) NOT NULL,

    -- Domain association
    domain VARCHAR(50),

    -- Constraints
    min_quality_score NUMERIC(5, 2),
    required_compliance TEXT[],
    forced_system_type VARCHAR(20),

    -- Truth Engine
    require_truth_engine BOOLEAN DEFAULT false,
    require_source_citation BOOLEAN DEFAULT false,
    max_ecd_threshold NUMERIC(4, 3),

CONSTRAINT weights_sum CHECK (ABS(
    weight_quality + weight_cost + weight_latency + weight_capability +
    weight_reliability + weight_compliance + weight_availability + weight_learning - 1.0
) < 0.01)
);

```

7.2 Domain Config Table

```

CREATE TABLE raws_domain_config (
    id VARCHAR(50) PRIMARY KEY, -- 7 domains
    weight_profile_id VARCHAR(50) REFERENCES raws_weight_profiles(id),
    min_quality_score NUMERIC(5, 2),
    max_ecd_threshold NUMERIC(4, 3),
    required_compliance TEXT[],
    forced_system_type VARCHAR(20),
    require_truth_engine BOOLEAN DEFAULT false,
    require_source_citation BOOLEAN DEFAULT false,
    detection_keywords TEXT[]
);

```

8. API Examples

8.1 Domain-Specific Selection

```

// Engineering domain
const result = await raws.select({
  requiredCapabilities: ['chat', 'function_calling'],
  estimatedInputTokens: 2000,
  estimatedOutputTokens: 1000,
  domain: 'engineering',
});

```

```

// Uses ENGINEERING profile: Q=0.30, K=0.20, C=0.15, L=0.15...

// Scientific domain
const result = await raws.select({
  requiredCapabilities: ['chat', 'reasoning'],
  estimatedInputTokens: 3000,
  estimatedOutputTokens: 2000,
  domain: 'scientific',
});
// Uses SCIENTIFIC profile: Q=0.35, K=0.20, E=0.07...

// Creative domain
const result = await raws.select({
  requiredCapabilities: ['chat', 'streaming'],
  estimatedInputTokens: 500,
  estimatedOutputTokens: 2000,
  domain: 'creative',
});
// Uses CREATIVE profile: C=0.25, L=0.20, E=0.10...

```

8.2 Auto-Detection

```

// Domain detected from query content
const result = await raws.select({
  requiredCapabilities: ['chat'],
  estimatedInputTokens: 1000,
  estimatedOutputTokens: 500,
  taskType: 'code_review', // Auto-maps to engineering domain
});

```

9. Testing

9.1 Profile Validation

```

describe('WeightProfiles', () => {
  it('should have 13 system profiles', () => {
    expect(Object.keys(WEIGHT_PROFILES)).toHaveLength(13);
  });

  it('all profiles should sum to 1.0', () => {
    for (const profile of Object.values(WEIGHT_PROFILES)) {
      const sum = Object.values(profile.weights).reduce((a, b) => a + b, 0);
      expect(sum).toBeCloseTo(1.0, 2);
    }
  });

  it('should map all 7 domains to profiles', () => {

```

```
    expect(Object.keys(DOMAIN_PROFILE_MAP)).toHaveLength(7);
  });
});
```

10. Performance

10.1 Latency Budget

Phase	Budget	Actual p99
Context + Weights	2ms	1.5ms
Filtering	5ms	4ms
Scoring (8 dim)	25ms	22ms
Ranking	15ms	12ms
Selection	3ms	2ms
Total	50ms	41.5ms

End of Engineering Documentation

Version 1.1.0 / January 2026