Contents

RADIANT AI Registry Seed Data System

Technical Documentation

Version: 4.18.1 | Last Updated: December 2024

Overview

The RADIANT Seed Data System manages versioned AI provider and model configurations that are used to populate the AI Registry during fresh installations. Seed data is stored separately from packages, can be versioned independently, and is selectable when building deployment packages.

Architecture

```
                        SEED DATA ARCHITECTURE


  config/seeds/
     registry.json          # Index of all seed versions
     v1/                    # Seed data version 1.0.0
        manifest.json      # Version metadata and stats
        providers.json     # 21 external providers
        external-models.json  # 50+ external models
        self-hosted-models.json  # 38 self-hosted models
        services.json      # 5 orchestration services
     v2/                    # Future seed versions...

  Build Time:

   build-package.sh    Select seed version    Include in package
   --seed-version 1


  Deploy Time (INSTALL only):

   DeploymentService   Read seeds from package    INSERT to database
   .executeInstall()
```

Critical Rules

Rule 1: NO HARDCODING IN DEPLOYER APP

The Swift Deployer app **MUST NOT** contain hardcoded lists of providers or models:

```swift
//  WRONG - Never do this
let providers = ["openai", "anthropic", "google", ...]

//  CORRECT - Fetch from Radiant API after deployment
let providers = try await radiantAPI.fetchProviders()
```

Rule 2: INSTALLER SEEDS, UPDATER PRESERVES

Mode	Seed Behavior
INSTALL	Seeds database with complete provider/model list
UPDATE	NEVER touches AI Registry - preserves admin customizations
ROLLBACK	Restores from snapshot - does not re-seed

Rule 3: ADMIN CONTROLS ALL

Everything in seed data is **editable by the administrator** post-deployment: - Enable/disable providers and models - Change pricing markup - Add new providers/models - Delete providers/models

---

Seed Data Structure

manifest.json

```json
{
  "version": "1.0.0",
  "name": "RADIANT AI Registry Seed Data",
  "description": "Complete provider and model seed data for fresh installations",
  "createdAt": "2024-12-25T00:00:00Z",
  "updatedAt": "2024-12-25T00:00:00Z",
  "compatibility": {
    "minRadiantVersion": "4.16.0",
    "maxRadiantVersion": "5.0.0"
  },
  "files": {
    "providers": "providers.json",
    "externalModels": "external-models.json",
    "selfHostedModels": "self-hosted-models.json",
    "services": "services.json"
  },
  "stats": {
    "externalProviders": 21,
    "externalModels": 50,
    "selfHostedModels": 38,
    "services": 5
  },
  "pricing": {
    "externalMarkup": 1.40,
    "selfHostedMarkup": 1.75
  }
}
```

providers.json

Each provider includes:

Field	Description
`id`	Unique identifier
`name`	Internal name
`displayName`	Human-readable name
`category`	Provider category (text_generation, image_generation, etc.)

Field	Description
`apiBaseUrl`	API endpoint
`authType`	Authentication type (bearer, api_key, iam)
`secretName`	AWS Secrets Manager path for API key
`features`	Supported features (streaming, vision, etc.)
`compliance`	Compliance certifications (SOC2, GDPR, HIPAA)
`rateLimit`	Rate limiting configuration

external-models.json

Each model includes:

Field	Description
`id`	Unique model identifier
`providerId`	Reference to provider
`modelId`	Provider's model ID
`litellmId`	LiteLLM routing ID
`category`	Model category
`capabilities`	Model capabilities
`contextWindow`	Max input tokens
`maxOutput`	Max output tokens
`pricing`	Cost per 1K tokens + markup
`minTier`	Minimum tier required

self-hosted-models.json

Each self-hosted model includes:

Field	Description
`id`	Unique model identifier
`instanceType`	SageMaker instance type
`thermal`	Thermal management config (COLD/WARM/HOT)
`license`	Open-source license
`pricing`	Hourly rate + per-unit pricing
`minTier`	Minimum tier required (typically 3+)

Building Packages with Seed Data

List Available Seed Versions

```
./tools/scripts/build-package.sh --list-seeds
```

Output:

```
Available Seed Data Versions:
  v1.0.0 - 21 providers, 50 external models, 38 self-hosted models
```

Build with Specific Seed Version

```bash
# Use default (latest) seed version
./tools/scripts/build-package.sh

# Use specific seed version
./tools/scripts/build-package.sh --seed-version 1
```

Package Manifest with Seed Data

The generated package manifest includes seed data information:

```json
{
  "schemaVersion": "2.1",
  "package": {
    "version": "4.18.1"
  },
  "seedData": {
    "version": "1.0.0",
    "hash": "abc123...",
    "externalProviders": 21,
    "externalModels": 50,
    "selfHostedModels": 38,
    "services": 5
  },
  "installBehavior": {
    "seedAIRegistry": true
  },
  "updateBehavior": {
    "seedAIRegistry": false
  }
}
```

Seed Data Categories

External Providers (21)

Category	Providers
Text Generation	OpenAI, Anthropic, Google, xAI, DeepSeek, Mistral, Cohere
Image Generation	OpenAI Images, Stability AI, FLUX
Video Generation	Runway, Luma AI
Audio	ElevenLabs, OpenAI Audio
Embeddings	OpenAI Embeddings, Voyage AI
Search	Perplexity
3D Generation	Meshy
Self-Hosted	SageMaker (internal)

External Models (50+)

Category	Example Models
Text	GPT-4o, Claude Sonnet 4, Gemini 2.0, Grok 3, DeepSeek R1
Reasoning	O1, O3 Mini, DeepSeek Reasoner
Code	Codestral
Image	DALL-E 3, Stable Diffusion 3, FLUX Pro
Video	Gen-3 Alpha, Ray 2
Audio	Whisper, TTS-1, Multilingual V2

Self-Hosted Models (38)

Category	Models
Vision Classification	EfficientNet, Swin Transformer, CLIP
Object Detection	YOLOv8 (Nano/Small/Medium/XLarge), Grounding DINO
Segmentation	SAM, SAM 2, MobileSAM
Speech	Whisper Large V3, Parakeet TDT
Scientific	AlphaFold 2, ESM-2
Medical	nnU-Net, MedSAM
Geospatial	Prithvi 100M/600M
3D	Nerfstudio
LLM	Mistral 7B, Llama 3 70B

Pricing Structure

External Providers

Default markup: **40% (1.40x)**

Example: GPT-4o - Provider cost: $0.0025/1K input, $0.01/1K output - Tenant cost: $0.0035/1K input, $0.014/1K output

Self-Hosted Models

Default markup: **75% (1.75x)**

Example: YOLOv8 Medium - Infrastructure cost: ~$2.47/hour + $0.005/image - Tenant cost: ~$4.32/hour + $0.00875/image

---

Creating New Seed Versions

1. Create Version Directory

```
mkdir config/seeds/v2
```

2. Create Required Files

- `manifest.json` - Version metadata
- `providers.json` - Provider definitions
- `external-models.json` - External model definitions
- `self-hosted-models.json` - Self-hosted model definitions
- `services.json` - Service definitions

3. Update Registry

Add new version to `config/seeds/registry.json`:

```
{
  "versions": [
    {
      "version": "2.0.0",
      "directory": "v2",
      "releaseDate": "2025-01-15",
      "status": "stable",
      "changelog": "Added new providers and models..."
    },
    // ... existing versions
  ]
}
```

4. Test Build

```
./tools/scripts/build-package.sh --seed-version 2
```

Database Seeding

During fresh installation, the DeploymentService generates SQL migrations from seed data:

```
-- Only runs if providers table is empty
DO $$
BEGIN
    IF NOT EXISTS (SELECT 1 FROM providers LIMIT 1) THEN
        -- Insert providers
        INSERT INTO providers (...) VALUES (...);

        -- Insert external models
        INSERT INTO models (...) VALUES (...);

        -- Insert self-hosted models
        INSERT INTO self_hosted_models (...) VALUES (...);
    END IF;
END $$;
```

Key behaviors: - Uses `ON CONFLICT DO NOTHING` to preserve admin changes - Only runs on fresh install (empty database) - Logs completion with model counts

---

Swift Service API

SeedDataService

```swift
actor SeedDataService {
    /// List available seed versions
    func listAvailableSeedVersions() async throws -> [SeedDataInfo]

    /// Load complete seed data for a version
    func loadSeedData(version: String) async throws -> SeedData

    /// Generate SQL migration from seed data
    func generateSeedMigration(seedData: SeedData) -> String
}
```

Usage in DeploymentService

```swift
func executeInstall(...) async throws -> DeploymentExecutionResult {
    // Load seed data from package
    let seedData = try await seedDataService.loadSeedData(
        version: package.manifest.seedData?.version ?? "1.0.0"
    )

    // Generate and run seed migration
    let seedSQL = seedDataService.generateSeedMigration(seedData: seedData)
    try await runMigration(sql: seedSQL)
}
```

Related Documentation

- Deployer Architecture - Deployment modes and package management
- Deployer Admin Guide - User-facing deployment documentation
- API Reference - Provider and model API endpoints