Contents

RADIANT v4.18.0 - Shared Types & Constants Export

Component: @radiant/shared Package **Language**: TypeScript **Files**: 60 type files, 41 constant files, 18 utility files **Purpose**: Single source of truth for types across all components

---

Architecture Narrative

The `@radiant/shared` package provides **type definitions and constants** used across all RADIANT components: Swift Deployer, CDK Infrastructure, Lambda handlers, and Admin Dashboard. This ensures type safety and consistency throughout the platform.

Package Structure

```
packages/shared/src/
  index.ts              # Main export
  types/               # 60 TypeScript type definition files
    index.ts          # Type exports
    brain-v6.types.ts
    cato.types.ts
    consciousness.types.ts
    ...
  constants/           # 41 constant definition files
    index.ts
    version.ts
    tiers.ts
    providers.ts
    ...
```

```
  utils/              # 18 utility function files
  errors/             # Error type definitions
  validation/         # Validation schemas
  templates/          # Template definitions
```

Core Constants

version.ts

Purpose: Single source of truth for version numbers.

```typescript
/**
 * RADIANT Version Constants
 * SINGLE SOURCE OF TRUTH for version numbers
 */

export const RADIANT_VERSION = '4.18.0';

export const VERSION = {
  major: 4,
  minor: 18,
  patch: 0,
  full: '4.18.0',
  build: process.env.BUILD_NUMBER || 'local',
  date: '2024-12',
} as const;

export const MIN_VERSIONS = {
  node: '20.0.0',
  npm: '10.0.0',
  cdk: '2.120.0',
  postgres: '15.0',
  swift: '5.9',
  macos: '13.0',
  xcode: '15.0',
} as const;

export const DOMAIN_PLACEHOLDER = 'YOUR_DOMAIN.com';

export function isDomainConfigured(domain: string): boolean {
  return !domain.includes(DOMAIN_PLACEHOLDER);
}
```

---

tiers.ts

Purpose: Infrastructure tier configurations with cost estimates and feature flags.

```typescript
/**
 * RADIANT v4.18.0 - Infrastructure Tiers
 * SINGLE SOURCE OF TRUTH
 */

import type { TierConfig, TierLevel, TierName } from '../types';

export const TIER_NAMES: Record<TierLevel, TierName> = {
  1: 'SEED',
  2: 'STARTUP',
  3: 'GROWTH',
  4: 'SCALE',
  5: 'ENTERPRISE',
};

export const TIER_CONFIGS: Record<TierLevel, TierConfig> = {
  1: {
    level: 1,
    name: 'SEED',
    description: 'Development and testing, minimal costs',
    vpcCidr: '10.0.0.0/20',
    azCount: 2,
    natGateways: 1,
    auroraMinCapacity: 0.5,
    auroraMaxCapacity: 2,
    enableGlobalDatabase: false,
    elasticacheNodes: 0,
    elasticacheNodeType: 'cache.t4g.micro',
    enableSelfHostedModels: false,
    maxSagemakerEndpoints: 0,
    litellmTaskCount: 1,
    litellmCpu: 256,
    litellmMemory: 512,
    enableWaf: false,
    enableGuardDuty: false,
    enableSecurityHub: false,
    enableHipaa: false,
    enableBrain: false,
    estimatedMonthlyCost: { min: 50, max: 150, typical: 85 },
  },
  2: {
    level: 2,
    name: 'STARTUP',
    description: 'Small production workloads',
    vpcCidr: '10.0.0.0/18',
    azCount: 2,
    natGateways: 1,
    auroraMinCapacity: 1,
```

```
  auroraMaxCapacity: 8,
  enableGlobalDatabase: false,
  elasticacheNodes: 1,
  elasticacheNodeType: 'cache.t4g.small',
  enableSelfHostedModels: false,
  maxSagemakerEndpoints: 0,
  litellmTaskCount: 2,
  litellmCpu: 512,
  litellmMemory: 1024,
  enableWaf: true,
  enableGuardDuty: true,
  enableSecurityHub: false,
  enableHipaa: false,
  enableBrain: false,
  estimatedMonthlyCost: { min: 200, max: 400, typical: 255 },
},
3: {
  level: 3,
  name: 'GROWTH',
  description: 'Medium production with self-hosted models',
  vpcCidr: '10.0.0.0/17',
  azCount: 3,
  natGateways: 2,
  auroraMinCapacity: 2,
  auroraMaxCapacity: 16,
  enableGlobalDatabase: false,
  elasticacheNodes: 2,
  elasticacheNodeType: 'cache.r6g.large',
  enableSelfHostedModels: true,
  maxSagemakerEndpoints: 10,
  litellmTaskCount: 3,
  litellmCpu: 1024,
  litellmMemory: 2048,
  enableWaf: true,
  enableGuardDuty: true,
  enableSecurityHub: true,
  enableHipaa: true,
  enableBrain: true,
  estimatedMonthlyCost: { min: 1000, max: 2500, typical: 1475 },
},
4: {
  level: 4,
  name: 'SCALE',
  description: 'Large production with multi-region',
  vpcCidr: '10.0.0.0/16',
  azCount: 3,
  natGateways: 3,
  auroraMinCapacity: 4,
```

```
    auroraMaxCapacity: 64,
    enableGlobalDatabase: true,
    elasticacheNodes: 3,
    elasticacheNodeType: 'cache.r6g.xlarge',
    enableSelfHostedModels: true,
    maxSagemakerEndpoints: 30,
    litellmTaskCount: 5,
    litellmCpu: 2048,
    litellmMemory: 4096,
    enableWaf: true,
    enableGuardDuty: true,
    enableSecurityHub: true,
    enableHipaa: true,
    enableBrain: true,
    estimatedMonthlyCost: { min: 4000, max: 8000, typical: 5450 },
  },
  5: {
    level: 5,
    name: 'ENTERPRISE',
    description: 'Enterprise-grade global deployment',
    vpcCidr: '10.0.0.0/14',
    azCount: 3,
    natGateways: 3,
    auroraMinCapacity: 8,
    auroraMaxCapacity: 128,
    enableGlobalDatabase: true,
    elasticacheNodes: 6,
    elasticacheNodeType: 'cache.r6g.2xlarge',
    enableSelfHostedModels: true,
    maxSagemakerEndpoints: 100,
    litellmTaskCount: 10,
    litellmCpu: 4096,
    litellmMemory: 8192,
    enableWaf: true,
    enableGuardDuty: true,
    enableSecurityHub: true,
    enableHipaa: true,
    enableBrain: true,
    estimatedMonthlyCost: { min: 15000, max: 35000, typical: 21500 },
  },
};

export function getTierConfig(tier: TierLevel): TierConfig {
  return TIER_CONFIGS[tier];
}

export function getTierName(tier: TierLevel): TierName {
  return TIER_NAMES[tier];
```

```
}

export function validateTierForEnvironment(tier: TierLevel, environment: string): void {
  if (environment === 'prod' && tier < 3) {
    throw new Error('Production requires Tier 3 (GROWTH) or higher');
  }
}

export function getFeatureFlagsForTier(tier: TierLevel) {
  const config = TIER_CONFIGS[tier];
  return {
    selfHostedModels: config.enableSelfHostedModels,
    multiRegion: config.enableGlobalDatabase,
    waf: config.enableWaf,
    guardDuty: config.enableGuardDuty,
    hipaaCompliance: config.enableHipaa,
    advancedAnalytics: tier >= 3,
    customBranding: tier >= 4,
    sla: tier >= 4,
  };
}
```

Core Types

brain-v6.types.ts

Purpose: AGI Brain system types including SOFAI routing, context budgeting, and flash facts.

```
/**
 * RADIANT v6.0.4 - AGI Brain Types
 * Project AWARE - Autonomous Wakefulness And Reasoning Engine
 */

// SOFAI System Levels
export type SystemLevel = 'system1' | 'system1.5' | 'system2';

export interface SofaiRoutingDecision {
  level: SystemLevel;
  confidence: number;            // 0-1
  reasoning: string;
  trust: number;                 // 1 - entropy
  domainRisk: number;
  computeCost: number;
  timestamp: Date;
}

export interface SofaiConfig {
```

```
  system2Threshold: number;      // Default: 0.5
  domainRisks: Record<string, number>;
  enableSystem1_5: boolean;
  maxSystem2Latency: number;
}

// Context Budget Types
export interface ContextBudget {
  systemCore: number;            // ~500 tokens
  complianceGuardrails: number;  // ~400 tokens
  flashFacts: number;            // ~200 tokens
  ghostTokens: number;           // ~64 tokens
  userMessage: number;
  memories: number;
  responseReserve: number;       // MINIMUM 1000 tokens
  totalInput: number;
  compressionApplied: boolean;
}

// Flash Fact Types
export type FlashFactType =
  | 'identity' | 'allergy' | 'medical'
  | 'preference' | 'constraint' | 'correction';

export interface FlashFact {
  id: string;
  userId: string;
  tenantId: string;
  fact: string;
  factType: FlashFactType;
  priority: 'critical' | 'high' | 'normal';
  status: 'pending_dream' | 'consolidated' | 'failed_retry';
  createdAt: Date;
  consolidatedAt: Date | null;
}

// Brain Request/Response
export interface BrainInferenceRequest {
  userId: string;
  tenantId: string;
  prompt: string;
  conversationHistory?: ConversationMessage[];
  domain?: string;
  forceSystemLevel?: SystemLevel;
  options?: BrainInferenceOptions;
}

export interface BrainInferenceResponse {
```

```
  response: string;
  systemLevel: SystemLevel;
  routingDecision: SofaiRoutingDecision;
  budget: ContextBudget;
  ghostUpdated: boolean;
  flashFactsDetected: FlashFactDetection;
  latencyMs: number;
  tokenUsage: { input: number; output: number; total: number };
  verification?: {
    passed: boolean;
    ecdScore: number;
    refinementAttempts: number;
    blocked: boolean;
  };
}

export interface ConversationMessage {
  role: 'user' | 'assistant' | 'system';
  content: string;
  timestamp?: Date;
}
```

---

cato.types.ts

Purpose: Genesis Cato Safety Architecture types with immutable safety invariants.

```
/**
 * RADIANT Genesis Cato Safety Architecture Types
 * Version: 2.3.1
 *
 * THREE-LAYER NAMING CONVENTION:
 * 1. CATO = The user-facing AI persona name
 * 2. GENESIS CATO = The safety architecture/system
 * 3. MOODS = Operating modes (Balanced, Scout, Sage, Spark, Guide)
 */

// IMMUTABLE SAFETY INVARIANTS - CANNOT be changed at runtime
export const CATO_INVARIANTS = {
  /** CBFs NEVER relax - shields stay UP */
  CBF_ENFORCEMENT_MODE: 'ENFORCE' as const,
  /** Gamma is NEVER boosted during recovery */
  GAMMA_BOOST_ALLOWED: false,
  /** Destructive actions require confirmation */
  AUTO_MODIFY_DESTRUCTIVE: false,
  /** Audit trail is append-only */
  AUDIT_ALLOW_UPDATE: false,
  AUDIT_ALLOW_DELETE: false,
```

```
} as const;

export const DEFAULT_PERSONA_NAME = 'balanced';
export const RECOVERY_PERSONA_NAME = 'scout';

// Persona (Mood) Types
export interface PersonaDrives {
  curiosity: number;    // 0-1
  achievement: number;  // 0-1
  service: number;      // 0-1
  discovery: number;    // 0-1
  reflection: number;   // 0-1
}

export interface PersonaVoice {
  formality: 'casual' | 'balanced' | 'formal';
  verbosity: 'concise' | 'balanced' | 'elaborate';
  emotionExpression: 'reserved' | 'moderate' | 'expressive';
  technicalLevel: 'simplified' | 'adaptive' | 'technical';
}

export interface Persona {
  id: string;
  name: string;
  displayName: string;
  description: string;
  scope: 'system' | 'tenant' | 'user';
  tenantId?: string;
  userId?: string;
  drives: PersonaDrives;
  derivedCMatrix?: CMatrix;
  defaultGamma: number;
  voice: PersonaVoice;
  presentation: PersonaPresentation;
  behavior: PersonaBehavior;
  isDefault: boolean;
  isActive: boolean;
}
```

Type File Inventory

Core Types (60 files)

File	Lines	Purpose
`admin-config.types.ts`	18K	Admin configuration types
`admin.types.ts`	3K	Admin user/role types

File	Lines	Purpose
agi-brain-plan.types.ts	9K	Brain planning types
agi-ideas.types.ts	7K	Idea management types
agi-orchestration.types.ts	10K	Orchestration types
ai.types.ts	3K	AI model types
app.types.ts	2K	Application types
aws-monitoring.types.ts	13K	AWS monitoring types
billing.types.ts	2K	Billing types
bipolar-rating.types.ts	8K	Rating system types
brain-v6.types.ts	9K	AGI Brain types
cato.types.ts	17K	Cato safety types
cognition.types.ts	10K	Cognition types
cognitive-architecture.types.ts	13K	Cognitive architecture
collaboration.types.ts	9K	Collaboration types
compliance-sandwich.types.ts	9K	Compliance sandwich
compliance.types.ts	2K	Compliance types
consciousness.types.ts	9K	Consciousness types
delight.types.ts	11K	Delight engine types
domain-ethics.types.ts	7K	Domain ethics types
domain-taxonomy.types.ts	10K	Domain taxonomy
dreaming.types.ts	7K	Dreaming types
ecd.types.ts	8K	ECD verification
environment.types.ts	1K	Environment types
experiments.types.ts	1K	A/B testing types
feedback.types.ts	7K	Feedback types
formal-reasoning.types.ts	22K	Formal reasoning
ghost.types.ts	7K	Ghost vector types
inference-components.types.ts	9K	Inference components
intelligence-aggregator.types.ts	10K	Intelligence aggregation
library-execution.types.ts	9K	Library execution
library-registry.types.ts	11K	Library registry
localization.types.ts	5K	i18n types
metrics-learning.types.ts	17K	Metrics learning
model-coordination.types.ts	8K	Model coordination
preprompt.types.ts	11K	Preprompt types
provider-rejection.types.ts	8K	Provider rejection
result-derivation.types.ts	11K	Result derivation
revenue.types.ts	8K	Revenue types
security.types.ts	8K	Security types
thinktank-generative-ui.types.ts	34K	Think Tank UI types
thinktank.types.ts	7K	Think Tank core types
translation-middleware.types.ts	16K	Translation middleware
user-registry.types.ts	13K	User registry
user-rules.types.ts	12K	User rules types

Constants (41 files)

File	Purpose
version.ts	Version constants
tiers.ts	Tier configurations
providers.ts	AI provider definitions
regions.ts	AWS region configs
environments.ts	Environment definitions
apps.ts	Application constants
brain.constants.ts	Brain configuration
cognition.constants.ts	Cognition constants
domain-ethics-registry.ts	Domain ethics registry
models/	Model configurations

Utilities (18 files)

File	Purpose
token-counting.ts	Token counting utilities
validation.ts	Input validation
formatting.ts	Output formatting
hashing.ts	Cryptographic hashing
date-utils.ts	Date manipulation
string-utils.ts	String utilities

Usage Examples

Importing Types

```
// Import from main package
import {
  RADIANT_VERSION,
  TierConfig,
  BrainInferenceRequest,
  SystemLevel,
  FlashFact,
} from '@radiant/shared';

// Import specific modules
import { TIER_CONFIGS } from '@radiant/shared/constants';
import { BrainInferenceResponse } from '@radiant/shared/types';
```

Type-Safe API Calls

```
import { BrainInferenceRequest, BrainInferenceResponse } from '@radiant/shared';

async function callBrain(request: BrainInferenceRequest): Promise<BrainInferenceResponse> {
  const response = await fetch('/api/brain/inference', {
```

```
    method: 'POST',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify(request),
  });
  return response.json();
}
```

Tier Configuration

```
import { getTierConfig, TierLevel } from '@radiant/shared';

function calculateInfrastructureCost(tier: TierLevel): number {
  const config = getTierConfig(tier);
  return config.estimatedMonthlyCost.typical;
}
```

This concludes the Shared Types export. See other export files for additional components.