# RADIANT API Reference

RADIANT Team

2025-12-28

## Contents

## RADIANT API Reference

Complete API documentation for the RADIANT AI Platform.

**Base URL:** `https://api.radiant.example.com/v2`

**Authentication:** Bearer token (API key or JWT)

```
Authorization: Bearer rad_your_api_key
```

---

## Chat Completions

### Create Chat Completion

`POST /v2/chat/completions`

Create a chat completion with any supported model.

**Request Body:**

| Field | Type | Required | Description |
|---|---|---|---|
| `model` | string | | Model ID (e.g., `gpt-4o`, `claude-3-sonnet`) |
| `messages` | array | | Array of message objects |
| `max_tokens` | integer | | Maximum tokens to generate |
| `temperature` | number | | Sampling temperature (0-2) |
| `top_p` | number | | Nucleus sampling (0-1) |
| `stream` | boolean | | Stream response tokens |
| `stop` | string/array | | Stop sequences |
| `functions` | array | | Function definitions |
| `function_call` | string/object | | Function calling mode |

**Message Object:**

| Field | Type | Required | Description |
|---|---|---|---|
| `role` | string | | `system`, `user`, `assistant`, `function` |
| `content` | string | | Message content |
| `name` | string | | Function name (for function messages) |

**Example Request:**

```json
{
  "model": "gpt-4o",
  "messages": [
    {"role": "system", "content": "You are a helpful assistant."},
    {"role": "user", "content": "Hello!"}
  ],
  "max_tokens": 1000,
  "temperature": 0.7
}
```

**Response:**

```json
{
  "id": "chatcmpl_abc123",
  "object": "chat.completion",
  "created": 1703980800,
  "model": "gpt-4o",
  "choices": [
    {
      "index": 0,
      "message": {
        "role": "assistant",
        "content": "Hello! How can I help you today?"
      },
      "finish_reason": "stop"
    }
  ],
  "usage": {
    "prompt_tokens": 25,
    "completion_tokens": 10,
    "total_tokens": 35
  }
}
```

---

## Models

### List Models

`GET /v2/models`

List all available models.

**Query Parameters:**

| Parameter | Type | Description |
|---|---|---|
| `category` | string | Filter by category (`chat`, `embedding`, `image`) |
| `provider` | string | Filter by provider (`openai`, `anthropic`, `google`) |

**Response:**

```json
{
  "object": "list",
  "data": [
    {
      "id": "gpt-4o",
      "object": "model",
      "created": 1703980800,
      "owned_by": "openai",
      "display_name": "GPT-4o",
      "category": "chat",
      "context_window": 128000,
      "input_cost_per_1k": 0.005,
      "output_cost_per_1k": 0.015,
      "capabilities": ["chat", "vision", "function_calling"]
    }
  ]
}
```

### Get Model

`GET /v2/models/{model_id}`

Get details for a specific model.

---

## Embeddings

### Create Embeddings

`POST /v2/embeddings`

Generate embeddings for text.

**Request Body:**

| Field | Type | Required | Description |
|---|---|---|---|
| `model` | string | | Embedding model ID |
| `input` | string/array | | Text to embed |
| `encoding_format` | string | | `float` or `base64` |

**Response:**

```
{
  "object": "list",
  "data": [
    {
      "object": "embedding",
      "index": 0,
      "embedding": [0.0023, -0.0091, ...]
    }
  ],
  "model": "text-embedding-3-small",
  "usage": {
    "prompt_tokens": 8,
    "total_tokens": 8
  }
}
```

---

## Billing

### Get Credit Balance

`GET /v2/billing/credits`

Get current credit balance.

**Response:**

```json
{
  "data": {
    "available": 150.50,
    "reserved": 10.00,
    "currency": "USD",
    "updated_at": "2024-01-15T10:30:00Z"
  }
}
```

### Get Usage

`GET /v2/billing/usage`

Get usage data for a period.

**Query Parameters:**

| Parameter | Type | Description |
|---|---|---|
| `start_date` | string | Start date (YYYY-MM-DD) |
| `end_date` | string | End date (YYYY-MM-DD) |
| `group_by` | string | `day`, `model`, `endpoint` |

---

## Webhooks

### List Webhooks

`GET /v2/webhooks`

List configured webhooks.

### Create Webhook

`POST /v2/webhooks`

**Request Body:**

```json
{
  "url": "https://your-server.com/webhook",
  "event_types": ["billing.low_balance", "usage.quota_reached"],
  "description": "Billing alerts"
}
```

**Response:**

```json
{
  "data": {
    "id": "wh_abc123",
    "url": "https://your-server.com/webhook",
    "secret": "whsec_xyz789...",
    "event_types": ["billing.low_balance", "usage.quota_reached"],
    "is_active": true,
    "created_at": "2024-01-15T10:30:00Z"
  }
}
```

### Test Webhook

`POST /v2/webhooks/{webhook_id}/test`

Send a test event to the webhook.

---

## Batch Processing

### Create Batch Job

`POST /v2/batch/jobs`

Create a batch processing job.

**Request Body:**

```json
{
  "type": "completions",
  "model": "gpt-4o",
  "input_file": "batch-input.jsonl",
  "options": {
    "system_prompt": "You are a helpful assistant.",
    "max_tokens": 500
  }
}
```

### Get Batch Job

`GET /v2/batch/jobs/{job_id}`

Get batch job status and results.

### List Batch Jobs

`GET /v2/batch/jobs`

List all batch jobs.

---

## Error Codes

RADIANT uses standardized error codes across all endpoints. See Error Codes Reference for the complete list.

### Error Categories

| Category | Code Range | Description |
|---|---|---|
| Authentication | `RADIANT_AUTH_1xxx` | Token, API key, session errors |
| Authorization | `RADIANT_AUTHZ_2xxx` | Permission, role, tenant errors |
| Validation | `RADIANT_VAL_3xxx` | Input validation errors |
| Resource | `RADIANT_RES_4xxx` | Not found, conflict, quota errors |
| Rate Limiting | `RADIANT_RATE_5xxx` | Throttling and rate limit errors |
| AI/Model | `RADIANT_AI_6xxx` | Model, provider, inference errors |
| Billing | `RADIANT_BILL_7xxx` | Credits, subscription errors |
| Storage | `RADIANT_STOR_8xxx` | File upload, storage errors |
| Internal | `RADIANT_INT_9xxx` | Server, database, timeout errors |

### Common Error Codes

| Code | HTTP | Retryable | Description |
|---|---|---|---|
| `RADIANT_AUTH_1001` | 401 | | Invalid authentication token |
| `RADIANT_AUTH_1004` | 401 | | Invalid API key |
| `RADIANT_VAL_3001` | 400 | | Required field missing |
| `RADIANT_RES_4001` | 404 | | Resource not found |
| `RADIANT_RATE_5001` | 429 | | Rate limit exceeded |
| `RADIANT_BILL_7001` | 402 | | Insufficient credits |
| `RADIANT_AI_6004` | 502 | | AI provider error |
| `RADIANT_INT_9001` | 500 | | Internal server error |

**Error Response Format:**

```json
{
  "error": {
    "code": "RADIANT_RATE_5001",
    "message": "Too many requests. Please slow down.",
    "category": "rate_limit",
    "retryable": true,
    "timestamp": "2024-12-25T10:30:00.000Z"
  }
}
```

**Retry-After Header:** Retryable errors include `Retry-After` header with seconds to wait.

---

## Rate Limits

| Tier | Requests/min | Tokens/min |
|---|---|---|
| Free | 10 | 10,000 |
| Starter | 50 | 50,000 |
| Professional | 100 | 200,000 |
| Business | 500 | 1,000,000 |
| Enterprise | 2,000 | Unlimited |

Rate limit headers:

```
X-RateLimit-Limit: 100
X-RateLimit-Remaining: 95
X-RateLimit-Reset: 1703980860
```

---

## SDKs

### TypeScript/JavaScript

```
npm install @radiant/sdk
```

```
import { RadiantClient } from '@radiant/sdk';

const client = new RadiantClient({ apiKey: 'your-key' });
const response = await client.chat.create({
  model: 'gpt-4o',
  messages: [{ role: 'user', content: 'Hello!' }],
});
```

### Python

```
pip install radiant-sdk
```

```
from radiant import RadiantClient

client = RadiantClient(api_key="your-key")
response = client.chat.create(
    model="gpt-4o",
    messages=[{"role": "user", "content": "Hello!"}],
)
```

### CLI

```
npm install -g @radiant/cli
radiant auth login
radiant chat send "Hello!"
```

---

### Changelog

See CHANGELOG.md for version history.

### Support

- **Email:** support@radiant.example.com
- **Documentation:** https://docs.radiant.example.com
- **Status:** https://status.radiant.example.com