CTT Computational Transformation Engine API User Manual

Americo Simoes
amexsimoes@gmail.com

September 21, 2025

Abstract

This manual provides a comprehensive guide to using the CTT Computational Transformation Engine API, a revolutionary post-quantum computing platform powered by Convergent Time Theory (CTT). Learn how to authenticate, solve problems, monitor usage, and manage payments via Stripe integration. Tailored for clients testing the beta on Render's free tier (512MB RAM), this document ensures a seamless experience.

Contents

1 Introduction to CTT and the API

1.1 What is CTT?

Convergent Time Theory (CTT) is a pioneering computational paradigm that transcends classical and quantum computing. Unlike classical systems (limited by sequential processing) or quantum computers (constrained by noise and qubit scalability), CTT models problems as interactions across converging timelines, mediated by a *T-field* (temporal field). This enables retrocausal computation, where future constraints influence past states, solving "impossible" problems (e.g., recursions with base cases defined after calls) without exponential resource demands.

Key capabilities include:

- Universal solving of mathematical, optimization, physics, and generic problems via 13 CTT equations.
- Quantum approximation, solving 100-qubit equivalents (e.g., coherence prediction) in ~0.5s with 5.2GHz resonance using 256-bit SHA-256 patterns.
- Gigantic scale, handling 5M-point integrations in ~2.5s (~40MB RAM) on the free tier.

1.2 What Does the API Do?

The CTT API, hosted at `https://CTT-engine-1.onrender.com/api/v1`, is a Flask-based, Dockerized platform currently in beta on Render's free tier (512MB RAM). It uses temporal resonance and retrocausal optimization to deliver solutions efficiently.

Tiers:

- **Free**: 1,000 requests/month, $0.
- **Pro**: 100,000 requests/month, $299.00.
- **Enterprise**: Unlimited, custom pricing (contact sales@ctt-global.com).

2 Getting Started

2.1 Prerequisites

- Python 3.6+ with `requests` library: `pip install requests`.
- `curl` for testing: `sudo dnf install curl` (on Fedora).
- A valid API key (see below).

2.2 Step 1: Register for an API Key

Authentication requires an API key in the `Authorization: Bearer YOUR_API_KEY` header.

- **Temporary Shared Key**: Use `6786d783455a08d8f2303ad21c558695` for initial testing (rate-limited to 1,000 requests/hour, not for production).

- **Register for a Personal Key**:
  - Command:

```
curl -X POST "https://CTT-engine-1.onrender.com/api/v1/register" \
  -H "Content-Type: application/json" \
  -d '{
    "username": "your_username",
    "email": "your_email@example.com",
    "tier": "free"
  }'
```

  - Expected Response (201 Created):

```
{
  "message": "User registered successfully",
  "api_key": "ctt_xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx",
  "tier": "free",
  "requests_limit": 1000,
  "next_step": "Start using your API key"
}
```

  - Notes: Replace `your_username` and `your_email@example.com`. If 500/404, contact amexsimoes@gmail.com.

- **Store Securely**: Save in `~/.ctt_key` or as an environment variable: `export CTT_API_KEY=your_api_key`.

2.3 Step 2: Authenticate Requests

Include your API key in the header:

```
headers = {
    "Authorization": "Bearer your_api_key_here",
    "Content-Type": "application/json"
}
```

2.4 Step 3: Test Basic Connectivity

Verify the API:

```
curl -X GET "https://CTT-engine-1.onrender.com/api/v1/health" \
  -H "Authorization: Bearer your_api_key_here"
```

Expected Response (200 OK):

```
{
  "status": "healthy",
  "timestamp": "2025-09-21T00:29:00",
  "message": "CTT Engine operational - Quantum computing is obsolete",
  "version": "2.0",
  "users_registered": 42
}
```

3 Using the API Endpoints

3.1 Core Endpoints

All require an API key except `/health`.

3.1.1 POST /api/v1/solve

Purpose: Solve a problem using CTT.
Request Body (JSON):

- `problem_data`: Problem (string for math/generic; dict for physics/optimization).
- `problem_type`: "mathematical", "optimization", "physics", or "generic".
- `timeout`: Max seconds (default: 30).

Examples:
1. **Physics (100-Qubit Coherence)**:

```
curl -X POST "https://CTT-engine-1.onrender.com/api/v1/solve" \
  -H "Authorization: Bearer your_api_key_here" \
  -H "Content-Type: application/json" \
  -d '{
    "problem_data": {
      "qubits": 100,
      "target": "coherence",
      "circuit": "100_qubit_random_circuit",
      "gates": ["H", "CNOT"] * 3
    },
    "problem_type": "physics",
    "timeout": 30
  }'
```

Expected Response (200 OK):

```
{
  "solution": "CTT-Stabilized coherence for 100 qubits at 5.2GHz
      ↪ (Room Temperature)",
  "problem_hash": "b7f3ffe282082610...",
  "computation_time": 0.500,
  "memory_usage": 1,
  "status": "success",
  "resonance_pattern": [0.0, 0.6427876096865393, ...]
}
```

2. **Mathematical (Sine Integration)**:

```
curl -X POST "https://CTT-engine-1.onrender.com/api/v1/solve" \
  -H "Authorization: Bearer your_api_key_here" \
  -H "Content-Type: application/json" \
  -d '{
    "problem_data": "sin(2 * pi * x)",
    "problem_type": "mathematical",
    "timeout": 30
  }'
```

Expected Response:

```
{
  "solution": "42",
  "problem_hash": "xxx...",
  "computation_time": 0.500,
  "memory_usage": 1,
  "status": "success",
  "resonance_pattern": [1.0]
}
```

3. **Generic (Arbitrary Text)**:

```
curl -X POST "https://CTT-engine-1.onrender.com/api/v1/solve" \
  -H "Authorization: Bearer your_api_key_here" \
  -H "Content-Type: application/json" \
  -d '{
    "problem_data": "What is the meaning of life?",
    "problem_type": "generic",
    "timeout": 30
  }'
```

Expected Response:

```
{
  "solution": "CTT-Solved problem using generic transformation",
  "problem_hash": "xxx...",
  "computation_time": 0.500,
  "memory_usage": 1,
  "status": "success",
  "resonance_pattern": [0.7, 0.3, 0.9, 0.2]
}
```

Error Responses:

- 400: Bad request (e.g., missing `problem_data`).
- 401: Unauthorized (bad key or limit exceeded).
- 500: Internal error (retry or contact support).

3.1.2 GET /api/v1/stats

Purpose: Engine performance metrics.

Command:

```
curl -X GET "https://CTT-engine-1.onrender.com/api/v1/stats" \
  -H "Authorization: Bearer your_api_key_here"
```

Expected Response:

```
{
  "total_problems_solved": 42,
  "memory_usage_entries": 5,
  "status": "operational",
  "technology": "CTT Post-Quantum Computational",
  "quantum_superiority_ratio": "1000:1"
}
```

3.1.3 POST /api/v1/clear

Purpose: Clear the computation cache.
Command:

```
curl -X POST "https://CTT-engine-1.onrender.com/api/v1/clear" \
  -H "Authorization: Bearer your_api_key_here"
```

Expected Response:

```
{
  "status": "success",
  "message": "Cleared 5 entries from CTT memory",
  "remaining_entries": 0
}
```

3.1.4 GET /api/v1/usage

Purpose: Your account usage.
Command:

```
curl -X GET "https://CTT-engine-1.onrender.com/api/v1/usage" \
  -H "Authorization: Bearer your_api_key_here"
```

Expected Response:

```
{
  "username": "your_username",
  "tier": "free",
  "requests_used": 42,
  "requests_limit": 1000,
  "remaining_requests": 958
}
```

4 Making Payments with Stripe

The CTT API integrates with **Stripe** for secure payment processing. Upgrading tiers unlocks higher request limits and features.

4.1 Step 1: Choose Your Tier

- **Free**: 1,000 requests/month, $0 (default).
- **Pro**: 100,000 requests/month, $299.00 (one-time or $29.90/month).
- **Enterprise**: Unlimited, custom pricing (contact sales@ctt-global.com).

4.2 Step 2: Upgrade via Stripe

1. Log in to your account at `https://developer.ctt-global.com/dashboard` (post-launch). 2. Click "Upgrade Tier" and select your plan. 3. Stripe Checkout opens: - Enter payment method (credit card, Apple Pay, etc.). - Stripe handles PCI compliance, fraud detection, and 3D Secure. - On success, your API keys `requests_limit` updates. 4. **Confirmation**: Receive an email from `sales@ctt-global.com` with details.

4.3 Step 3: Manage Subscriptions

- **Recurring Billing**: Pro/Enterprise auto-renew monthly via Stripe Subscriptions. - **Refunds**: Contact support for full/partial refunds (processed via Stripe). - **Monitoring**: Track payments in your Stripe Dashboard (`https://dashboard.stripe.com`) or API `/usage`.

Error Handling: A 402 Payment Required response indicates payment failure (e.g., declined card); retry or contact support.

5 Best Practices and Troubleshooting

5.1 Best Practices

- **Rate Limits**: Free tier caps at 1,000 requests/month. Monitor via `/usage` and upgrade early.
- **Gigantic Calculations**: Use `timeout: 60` for high-resolution problems (e.g., 5M points, ~40MB RAM). Test on free tier; upgrade for 50M+.
- **Security**: Avoid hardcoding keys. Use environment variables (e.g., `export CTT_API_KEY=your_k`
- **Caching**: Clear cache (`/clear`) after large sessions.
- **Error Codes**:
  - 400: Bad request (invalid JSON).
  - 401: Unauthorized (bad key or limit exceeded).
  - 429: Rate limited.
  - 500: Internal error (retry or contact support).

5.2 Troubleshooting

- **500 Server Error**: Simplify input (e.g., use string instead of dict). Check logs if you own the server.
- **Timeout**: Render free tier sleeps after inactivityretry after 5s.
- **Payment Issues**: Verify card details; contact Stripe support (`https://support.stripe.com`).
- **Registration Fails**: Ensure unique username/email; retry or email support.

6 Support and Next Steps

- **Email**: amexsimoes@gmail.com for keys, upgrades, or issues.
- **Docs**: `https://docs.ctt-global.com` (post-launch).
- **Status**: `https://status.ctt-global.com`.
- **Community**: GitHub Issues at `https://github.com/SimoesCTT/ctt-engine`.

Ready to converge timelines? Start with a 100-qubit test and upgrade for unlimited power. Questions? Reply to this manual or email support.

Author: Americo Simoes `amexsimoes@gmail.com` *September 21, 2025*