Official Robotics AI Cells Client Manual

Table of Contents

Introduction

The Robotics AI Cells system provides a collection of specialized AI-powered functions ("cells") for robotics applications. This manual explains how to access these cells programmatically using your API key.

Getting Started

1. **Register for an API Key**:
   - Visit the web interface at https://aemiliotis.github.io/Robotics-AI-Cells
   - Click "Register" and create an account
   - Your API key will be displayed after registration
2. **Base API URL**:
   - Production: `https://robotics-ai-cells-m5gy.onrender.com`

Authentication

All requests require: - `X-API-Key` header with your API key - `X-Session-Id` header with a unique session identifier - `X-Request-Id` header with a unique request identifier

```
POST /ai-api
Headers:
  X-API-Key: your_api_key_here
  X-Session-Id: unique_session_id
  X-Request-Id: unique_request_id
```

API Usage

Request Format

```
{
  "cells": ["cell_name1", "cell_name2"],
  "data": {
    "cell_name1": {
      "input_param1": "value1",
      "input_param2": "value2"
    }
  }
}
```

Response Format

```
{
  "success": true,
  "results": {
    "cell_name1": {
      "output_param1": "value1",
      "output_param2": "value2"
    }
  },
  "metadata": {
    "user_id": "123",
    "session_id": "session_123",
    "cell_count": 1
  }
}
```

Available Cells

See full list in web interface

Programming Examples

Python Example (PID Controller)

```
import requests
import uuid
import json

# Configuration
API_URL = "https://robotics-ai-cells-m5gy.onrender.com/ai-api"
API_KEY = "your_api_key_here"  # Replace with your actual API key

# Generate unique IDs
```

```python
session_id = str(uuid.uuid4())
request_id = str(uuid.uuid4())

# PID Controller parameters
pid_inputs = {
    "error": 0.5,        # Current error from setpoint
    "integral": 0.2,     # Accumulated integral term
    "last_error": 0.4    # Previous error
}

# Prepare request
headers = {
    "X-API-Key": API_KEY,
    "X-Session-Id": session_id,
    "X-Request-Id": request_id,
    "Content-Type": "application/json"
}

payload = {
    "cells": ["pid_controller"],
    "data": {
        "pid_controller": pid_inputs
    }
}

try:
    # Make API request
    response = requests.post(API_URL, headers=headers, json=payload)
    response.raise_for_status()

    # Process response
    result = response.json()
    if result.get("success"):
        pid_output = result["results"]["pid_controller"]
        print("PID Output:", pid_output)
        print("Execution Metadata:", result["metadata"])
    else:
        print("Error:", result.get("error", "Unknown error"))

except requests.exceptions.RequestException as e:
    print("API Request Failed:", str(e))
except json.JSONDecodeError:
    print("Failed to parse API response")
except KeyError:
    print("Unexpected response format")
```

JavaScript Example (PID Controller)

```javascript
const API_URL = "https://robotics-ai-cells-m5gy.onrender.com/ai-api";
const API_KEY = "your_api_key_here"; // Replace with your actual API key

// Generate unique IDs
const sessionId = crypto.randomUUID();
const requestId = crypto.randomUUID();

// PID Controller parameters
const pidInputs = {
    error: 0.5,        // Current error from setpoint
    integral: 0.2,     // Accumulated integral term
    last_error: 0.4    // Previous error
};

// Prepare request
const headers = {
    'X-API-Key': API_KEY,
    'X-Session-Id': sessionId,
    'X-Request-Id': requestId,
    'Content-Type': 'application/json'
};

const payload = {
    cells: ['pid_controller'],
    data: {
        pid_controller: pidInputs
    }
};

// Make API request
fetch(API_URL, {
    method: 'POST',
    headers: headers,
    body: JSON.stringify(payload)
})
.then(response => {
    if (!response.ok) {
        throw new Error(`HTTP error! status: ${response.status}`);
    }
    return response.json();
})
.then(data => {
    if (data.success) {
        console.log("PID Output:", data.results.pid_controller);
```

```
        console.log("Execution Metadata:", data.metadata);
    } else {
        console.error("API Error:", data.error || "Unknown error");
    }
})
.catch(error => {
    console.error("Request Failed:", error);
});
```

Error Handling

Common error responses:

HTTP Code	Error	Description
400	Invalid request	Missing required parameters or invalid format
401	Unauthorized	Invalid or missing API key
429	Rate limit exceeded	Too many requests
500	Server error	Internal server error

Best Practices

1. **Session Management**:
   - Generate a new session ID for each user session
   - Reuse the same session ID for all requests in a single session
   - Sessions automatically expire after 24 hours of inactivity
2. **Request IDs**:
   - Generate a unique request ID for each API call
   - Useful for debugging and tracking specific requests
3. **Performance**:
   - Batch multiple cell executions in a single request when possible
   - Cache results when appropriate
   - Implement retry logic for transient failures
4. **Security**:
   - Never expose your API key in client-side code
   - Rotate API keys periodically
   - Use HTTPS for all requests

Rate Limits

- **200 requests per day** per API key
- **50 requests per hour** per API key
- **10 concurrent requests** per user

Troubleshooting

Problem: Getting 401 Unauthorized errors
Solution:
- Verify your API key is correct - Check that the `X-API-Key` header is being sent - Ensure your account is still active

Problem: Cells not returning expected results
Solution:
- Verify input parameters match the expected format - Check the web interface for the latest cell documentation - Test with simple inputs to isolate the issue

Problem: Slow response times
Solution:
- Check your network connection - Verify the API status at `/ping` - Reduce batch size if making large requests

For additional support, contact: etion.prosfores@gmail.com

Documentation version: 1.1
Last updated: 2025-7-29