Apidocumentation

Source File: generated-documents\technical-design\apidocumentation.md
Generated: 08/07/2025 at 09:43:57
Generated by: Requirements Gathering Agent - PDF Converter

APIDocumentation

Generated by adpa-enterprise-framework-automation v3.1.6
Category: technical-design
Generated: 2025-07-05T17:02:57.110Z
Description:

Self-Charging Electric Vehicle (SCEV) API Documentation

This document outlines the API for interacting with the Self-Charging Electric Vehicle (SCEV) system. This API allows developers to access real-time data from the vehicle, control certain aspects of energy management, and receive notifications. This is a preliminary API specification and may be subject to change.

1. API Overview

This RESTful API utilizes JSON for data exchange and employs standard HTTP methods (GET, POST, PUT, DELETE). It provides access to various aspects of the SCEV's energy harvesting and management system. The API is designed for secure, scalable, and reliable access to vehicle data and control functions.

2. Authentication Methods

API access requires authentication using a JSON Web Token (JWT). The JWT is obtained by providing valid vehicle credentials (VIN and a pre-shared secret) to the `/auth` endpoint.

3. Endpoint Definitions

All endpoints are prefixed with `/api/v1/`.

Method	Endpoint	Description	Request Body	Response Body
POST	`/auth`	Authenticate and obtain a JWT.	`{"vin": "VIN", "secret": "SECRET"}`	`{"token": "JWT"}`
GET	`/vehicle/status`	Retrieve the current status of the vehicle (battery level, energy generation rates, location).	None	See Response Schema (Status)
GET	`/energy/history`	Retrieve historical energy generation and consumption data.	`{"from": timestamp, "to": timestamp}`	See Response Schema (History)
PUT	`/settings/drivingMode`	Set the vehicle's driving mode (e.g., Eco, Sport).	`{"mode": "Eco"}`	`{"status": "success"}`
GET	`/energy/sources`	Get real-time energy generation data from each source (solar,	None	See Response Schema (Sources)

Method	Endpoint	Description	Request Body	Response Body
		kinetic, thermal).		

4. Request/Response Formats

All requests and responses use JSON.

Response Schema (Status):

```
{
  "batteryLevel": 75, // Percentage
  "solarPower": 100, // Watts
  "kineticPower": 50, // Watts
  "thermalPower": 20, // Watts
  "location": {
    "latitude": 34.0522,
    "longitude": -118.2437
  },
  "timestamp": "2024-10-27T10:30:00Z"
}
```

Response Schema (History):

```
{
  "data": [
    {"timestamp": "2024-10-27T10:00:00Z", "generated": 500, "consumed": 300},
    {"timestamp": "2024-10-27T10:15:00Z", "generated": 400, "consumed": 250}
    // ... more data points
  ]
}
```

Response Schema (Sources):

```
{
  "solar": 120, // Watts
  "kinetic": 60, // Watts
  "thermal": 30  // Watts
}
```

5. Error Codes

Code	Description
400	Bad Request (invalid input)
401	Unauthorized (invalid JWT)
403	Forbidden (insufficient permissions)
404	Not Found (resource not found)
500	Internal Server Error
429	Too Many Requests (rate limiting exceeded)

6. Rate Limiting

The API implements rate limiting to prevent abuse. The specific limits will depend on the subscription plan. Responses with a 429 status code will include details on the rate limit and retry-after time.

7. Versioning Strategy

API versioning is handled through URI versioning (e.g., `/api/v1/`, `/api/v2/`).

8. Security Considerations

- **HTTPS:** All communication should be done over HTTPS.
- **JWT Authentication:** Securely store and manage JWTs.
- **Input Validation:** All input is validated to prevent injection attacks.
- **Rate Limiting:** Protects against denial-of-service attacks.

9. Example Usage (Python)

```
import requests
import json

# Obtain JWT
auth_url = "https://api.scevsystem.com/api/v1/auth"
auth_data = {"vin": "YOUR_VIN", "secret": "YOUR_SECRET"}
auth_response = requests.post(auth_url, json=auth_data)
jwt = auth_response.json()["token"]

# Get vehicle status
headers = {"Authorization": f"Bearer {jwt}"}
```

```
status_url = "https://api.scevsystem.com/api/v1/vehicle/status"
status_response = requests.get(status_url, headers=headers)
print(json.dumps(status_response.json(), indent=2))
```

10. Testing Guidelines

Use a tool like Postman or curl to test the API endpoints. Ensure to replace placeholder values with your actual VIN, secret, and other relevant data. Thorough testing should cover all endpoints, including error handling and boundary conditions.

This documentation provides a foundation for interacting with the SCEV API. Further details and specific implementation notes will be provided in subsequent releases. Contact support@scevsystem.com for any questions or issues.
