Ultimate Lomanco CRM Automation System - API Documentation

Table of Contents

Overview

The Ultimate Lomanco CRM Automation System provides a comprehensive RESTful API for managing customer leads, job counts, and automated vent calculations. The API supports real-time data synchronization, advanced caching, and enterprise-grade security features.

Key Features

- ☑ Real-time lead and job count management
- ☑ Automated Lomanco vent calculations
- ☑ Batch processing capabilities
- ☑ Advanced caching and performance optimization
- ☑ Comprehensive monitoring and analytics
- ☑ Enterprise security and audit trails

Authentication

The API uses Google Apps Script authentication with session-based security.

Authentication Headers

```
Content-Type: application/json
Authorization: Bearer YOUR_ACCESS_TOKEN
X-Session-ID: SESSION_IDENTIFIER
```

Session Management

```
// Initialize session
const session = await securityManager.generateSessionToken(userId);

// Validate session
const isValid = securityManager.validateSession(sessionToken);
```

Base URL & Configuration

Production Environment

```
Base URL: https://script.google.com/macros/s/YOUR_SCRIPT_ID/exec
```

Development Environment

```
Base URL: https://script.google.com/macros/s/YOUR_DEV_SCRIPT_ID/dev
```

Configuration

```
const config = {
  spreadsheetId: '1E9VX7XI7GNGJa8Hq9__XAWv95WNgfJto1gUEXMbGqi0',
  leadsSheet: 'Bhotchleads',
  jobCountSheet: 'Job Count',
  timeout: 30000,
  retryAttempts: 3
};
```

Core API Endpoints

Leads Management

GET /leads

Retrieve all customer leads with optional filtering and pagination.

Parameters:

- `limit` (integer, optional): Number of results per page (default: 50)

- `offset` (integer, optional): Number of results to skip (default: 0)
- `status` (string, optional): Filter by lead status
- `search` (string, optional): Search term for name, email, or phone

Example Request:

```
GET /leads?limit=20&offset=0&status=active&search=john
```

Example Response:

```
{
  "success": true,
  "message": "Leads retrieved successfully",
  "timestamp": "2024-01-15T10:30:00.000Z",
  "data": [
    {
      "id": "lead_123",
      "name": "John Smith",
      "email": "john@example.com",
      "phone": "555-0123",
      "address": "123 Main St, City, State 12345",
      "status": "active",
      "createdAt": "2024-01-10T09:00:00.000Z",
      "updatedAt": "2024-01-15T10:30:00.000Z"
    }
  ],
  "pagination": {
    "total": 150,
    "limit": 20,
    "offset": 0,
    "hasMore": true
  }
}
```

POST /leads

Create a new customer lead.

Request Body:

```
{
  "name": "Jane Doe",
  "email": "jane@example.com",
  "phone": "555-0456",
  "address": "456 Oak Ave, City, State 12345",
  "notes": "Interested in ridge vents",
  "status": "new"
}
```

Example Response:

```
{
  "success": true,
  "message": "Lead created successfully",
  "timestamp": "2024-01-15T10:35:00.000Z",
  "data": {
    "id": "lead_124",
    "name": "Jane Doe",
    "email": "jane@example.com",
    "phone": "555-0456",
    "address": "456 Oak Ave, City, State 12345",
    "notes": "Interested in ridge vents",
    "status": "new",
    "createdAt": "2024-01-15T10:35:00.000Z",
    "updatedAt": "2024-01-15T10:35:00.000Z"
  }
}
```

PUT /leads/{id}

Update an existing lead.

Parameters:

- `id` (string, required): Lead identifier

Request Body:

```
{
  "status": "contacted",
  "notes": "Called customer, scheduled visit for next week"
}
```

DELETE /leads/{id}

Delete a lead (soft delete with audit trail).

Parameters:

- `id` (string, required): Lead identifier

Job Counts Management

GET /job-counts

Retrieve all job counts with calculation results.

Example Response:


```
{
  "success": true,
  "message": "Job counts retrieved successfully",
  "timestamp": "2024-01-15T10:40:00.000Z",
  "data": [
    {
      "id": "job_456",
      "customerName": "Smith Residence",
      "address": "789 Pine St, City, State 12345",
      "sqft": 2500,
      "ridgeVents": 12,
      "turbineVents": 8,
      "rimeFlow": 1875.00,
      "calculationMethod": "automated",
      "calculatedAt": "2024-01-15T10:40:00.000Z",
      "createdAt": "2024-01-14T14:20:00.000Z",
      "updatedAt": "2024-01-15T10:40:00.000Z"
    }
  ]
}
```


POST /job-counts

Create a new job count entry.

Request Body:


```
{
  "customerName": "Johnson Home",
  "address": "321 Elm St, City, State 12345",
  "sqft": 3200,
  "notes": "Large residential project"
}
```


PUT /job-counts/{id}

Update an existing job count.

Automation Endpoints

Lomanco Vent Calculations

POST /calculate-lomanco-vents

Perform automated vent calculations using the Lomanco system.

Request Body:


```
{
  "sqft": 2800,
  "method": "automated",
  "fallbackEnabled": true
}
```


Example Response:


```
{
  "success": true,
  "message": "Vent calculation completed successfully",
  "timestamp": "2024-01-15T11:00:00.000Z",
  "data": {
    "sqft": 2800,
    "ridgeVents": 14,
    "turbineVents": 9,
    "rimeFlow": 2100.00,
    "method": "web_automation",
    "calculationTime": 3.2,
    "fallbackUsed": false,
    "confidence": 0.98
  },
  "metadata": {
    "calculationId": "calc_789",
    "sourceUrl": "https://ventselector.lomanco.com",
    "processedAt": "2024-01-15T11:00:00.000Z",
    "validationPassed": true
  }
}
```

POST /batch-calculate-vents

Perform batch calculations for multiple job counts.

Request Body:

```
{
  "jobCountIds": ["job_456", "job_457", "job_458"],
  "options": {
    "method": "automated",
    "fallbackEnabled": true,
    "maxConcurrent": 3
  }
}
```

Example Response:

```
{
  "success": true,
  "message": "Batch calculation completed",
  "timestamp": "2024-01-15T11:05:00.000Z",
  "data": {
    "totalJobs": 3,
    "successful": 3,
    "failed": 0,
    "processingTime": 8.7,
    "results": [
      {
        "jobCountId": "job_456",
        "success": true,
        "ridgeVents": 12,
        "turbineVents": 8,
        "rimeFlow": 1875.00,
        "method": "web_automation"
      }
    ]
  }
}
```

Fallback Calculations

POST /calculate-mathematical-fallback

Use mathematical formulas when web automation fails.

Request Body:

```
{
  "sqft": 2500,
  "ventilationRatio": 300,
  "ridgeVentNFA": 18,
  "turbineVentNFA": 140
}
```

Monitoring Endpoints

System Health

GET /health

Get overall system health status.

Example Response:

```
{
  "success": true,
  "data": {
    "status": "healthy",
    "timestamp": "2024-01-15T11:10:00.000Z",
    "components": {
      "database": "healthy",
      "cache": "healthy",
      "automation": "healthy",
      "notifications": "healthy"
    },
    "metrics": {
      "uptime": 2592000,
      "responseTime": 245,
      "successRate": 0.987,
      "errorRate": 0.013
    }
  }
}
```

GET /metrics

Get detailed performance metrics.

Parameters:

- `timeRange` (integer, optional): Time range in milliseconds (default: 3600000)
- `category` (string, optional): Metric category filter

Example Response:


```
{
  "success": true,
  "data": {
    "timeRange": 3600000,
    "period": {
      "start": "2024-01-15T10:10:00.000Z",
      "end": "2024-01-15T11:10:00.000Z"
    },
    "performance": {
      "averageResponseTime": 1.8,
      "p95ResponseTime": 4.2,
      "successRate": 0.987,
      "totalRequests": 1547
    },
    "automation": {
      "calculationsPerformed": 234,
      "successRate": 0.954,
      "averageCalculationTime": 3.1,
      "fallbackUsage": 0.046
    },
    "cache": {
      "hitRate": 0.834,
      "memoryUsage": 0.67,
      "evictions": 12
    }
  }
}
```


System Monitoring

GET /alerts

Retrieve active system alerts.

Example Response:


```
{
  "success": true,
  "data": {
    "active": [
      {
        "id": "alert_123",
        "type": "performance",
        "severity": "warning",
        "message": "Response time above threshold",
        "timestamp": "2024-01-15T11:05:00.000Z",
        "details": {
          "metric": "responseTime",
          "value": 5.2,
          "threshold": 5.0
        }
      }
    ],
    "resolved": 45,
    "total": 46
  }
}
```

Security Endpoints

Security Validation

POST /validate-input

Validate and sanitize input data.

Request Body:

```
{
  "data": {
    "name": "John <script>alert('xss')</script> Smith",
    "email": "john@example.com",
    "sqft": "2500"
  },
  "types": {
    "name": "name",
    "email": "email",
    "sqft": "sqft"
  }
}
```

Example Response:

```
{
  "success": true,
  "data": {
    "sanitized": {
      "name": "John  Smith",
      "email": "john@example.com",
      "sqft": 2500
    },
    "warnings": [
      "Potentially malicious content removed from name field"
    ]
  }
}
```

Session Management

POST /sessions

Create a new session.

Request Body:

```
{
  "userId": "user_123",
  "deviceInfo": {
    "userAgent": "Mozilla/5.0...",
    "platform": "web"
  }
}
```

DELETE /sessions/{sessionId}

Invalidate a session.

⚠ Error Handling

Standard Error Response Format

```
{
  "success": false,
  "error": {
    "code": "VALIDATION_ERROR",
    "message": "Invalid input data provided",
    "details": {
      "field": "sqft",
      "reason": "Value must be between 1 and 100,000"
    },
    "timestamp": "2024-01-15T11:15:00.000Z",
    "requestId": "req_abc123"
  }
}
```

Error Codes

Code	Description	HTTP Status
VALIDATION_ERROR	Input validation failed	400
AUTHENTICATION_ERROR	Authentication required	401
AUTHORIZATION_ERROR	Insufficient permissions	403
NOT_FOUND	Resource not found	404
RATE_LIMIT_EXCEEDED	Too many requests	429
AUTOMATION_FAILED	Calculation automation failed	500
SYSTEM_ERROR	Internal system error	500
SERVICE_UNAVAILABLE	External service unavailable	503

🚦 Rate Limiting

Rate Limits

- **Standard API calls**: 60 requests per minute
- **Calculation endpoints**: 20 requests per minute
- **Batch operations**: 5 requests per minute
- **Monitoring endpoints**: 100 requests per minute

Rate Limit Headers

```
X-RateLimit-Limit: 60
X-RateLimit-Remaining: 45
X-RateLimit-Reset: 1610712600
X-RateLimit-RetryAfter: 15
```

Response Formats

Success Response

```
{
  "success": true,
  "message": "Operation completed successfully",
  "timestamp": "2024-01-15T11:20:00.000Z",
  "data": { /* Response data */ },
  "metadata": { /* Additional metadata */ }
}
```

Pagination Response

```
{
  "success": true,
  "data": [ /* Array of results */ ],
  "pagination": {
    "total": 500,
    "limit": 50,
    "offset": 100,
    "hasMore": true,
    "nextOffset": 150
  }
}
```

Code Examples

JavaScript/Node.js

Initialize Client

```
import { UltimateCRMClient } from './ultimate-crm-client';

const client = new UltimateCRMClient({
  baseUrl: 'https://script.google.com/macros/s/YOUR_SCRIPT_ID/exec',
  apiKey: 'your-api-key',
  timeout: 30000
});
```

Create Lead

```
const newLead = await client.leads.create({
  name: 'John Smith',
  email: 'john@example.com',
  phone: '555-0123',
  address: '123 Main St, City, State 12345'
});

console.log('Lead created:', newLead.data.id);
```

Perform Calculation

```
const calculation = await client.automation.calculateVents({
  sqft: 2500,
  method: 'automated',
  fallbackEnabled: true
});

console.log('Calculation result:', calculation.data);
```

Batch Processing

```
const batchResult = await client.automation.batchCalculate({
  jobCountIds: ['job_1', 'job_2', 'job_3'],
  options: {
    maxConcurrent: 2,
    fallbackEnabled: true
  }
});

console.log(`Processed ${batchResult.data.successful}/${batchResult.data.totalJobs} jobs`);
```

React Integration

React Hook for API Calls

```
import { useState, useEffect } from 'react';
import { useUltimateCRM } from './hooks/useUltimateCRM';

function LeadsManager() {
  const { client } = useUltimateCRM();
  const [leads, setLeads] = useState([]);
  const [loading, setLoading] = useState(true);

  useEffect(() => {
    const fetchLeads = async () => {
      try {
        const response = await client.leads.getAll();
        setLeads(response.data);
      } catch (error) {
        console.error('Failed to fetch leads:', error);
      } finally {
        setLoading(false);
      }
    };

    fetchLeads();
  }, [client]);

  const handleCreateLead = async (leadData) => {
    try {
      const newLead = await client.leads.create(leadData);
      setLeads(prev => [...prev, newLead.data]);
    } catch (error) {
      console.error('Failed to create lead:', error);
    }
  };

  return (
    <div>
      {loading ? (
        <div>Loading leads...</div>
      ) : (
        <LeadsList leads={leads} onCreateLead={handleCreateLead} />
      )}
    </div>
  );
}
```

Python Integration

Python Client Example

```python
import requests
import json
from typing import Dict, List, Optional

class UltimateCRMClient:
    def __init__(self, base_url: str, api_key: str):
        self.base_url = base_url
        self.headers = {
            'Content-Type': 'application/json',
            'Authorization': f'Bearer {api_key}'
        }

    def create_lead(self, lead_data: Dict) -> Dict:
        """Create a new lead"""
        response = requests.post(
            f"{self.base_url}/leads",
            headers=self.headers,
            json=lead_data
        )
        response.raise_for_status()
        return response.json()

    def calculate_vents(self, sqft: int, method: str = 'automated') -> Dict:
        """Calculate vent requirements"""
        payload = {
            'sqft': sqft,
            'method': method,
            'fallbackEnabled': True
        }

        response = requests.post(
            f"{self.base_url}/calculate-lomanco-vents",
            headers=self.headers,
            json=payload
        )
        response.raise_for_status()
        return response.json()

# Usage example
client = UltimateCRMClient(
    base_url='https://script.google.com/macros/s/YOUR_SCRIPT_ID/exec',
    api_key='your-api-key'
)

# Create a lead
lead = client.create_lead({
    'name': 'Jane Doe',
    'email': 'jane@example.com',
    'phone': '555-0456'
})

# Calculate vents
calculation = client.calculate_vents(sqft=2500)
print(f"Ridge vents needed: {calculation['data']['ridgeVents']}")
```

🔧 SDKs & Integrations

Official SDKs

- **JavaScript/TypeScript**: `@ultimate-crm/javascript-sdk`
- **Python**: `ultimate-crm-python`
- **PHP**: `ultimate-crm/php-sdk`

Third-Party Integrations

- **Zapier**: Connect with 3,000+ apps
- **Webhooks**: Real-time event notifications
- **REST API**: Standard HTTP integration
- **GraphQL**: Coming soon

Installation

NPM

```bash
npm install @ultimate-crm/javascript-sdk
```

Python

```bash
pip install ultimate-crm-python
```

PHP

```bash
composer require ultimate-crm/php-sdk
```

Performance Guidelines

Best Practices

1. **Use batch operations** for multiple calculations
2. **Implement client-side caching** for frequently accessed data
3. **Use pagination** for large datasets
4. **Handle rate limits** with exponential backoff
5. **Monitor response times** and optimize accordingly

Caching Strategy

```
// Example caching implementation
const cache = new Map();
const CACHE_TTL = 5 * 60 * 1000; // 5 minutes

async function getCachedLeads() {
  const cacheKey = 'leads_all';
  const cached = cache.get(cacheKey);

  if (cached && Date.now() - cached.timestamp < CACHE_TTL) {
    return cached.data;
  }

  const leads = await client.leads.getAll();
  cache.set(cacheKey, {
    data: leads,
    timestamp: Date.now()
  });

  return leads;
}
```

Testing & Development

API Testing

Use the provided test suite to validate API functionality:

```
npm test -- --grep "API Integration"
```

Mock Server

For development, use the mock server:

```
npm run mock-server
```

Postman Collection

Import the provided Postman collection for easy API testing:

- Download Collection (./postman/ultimate-crm-api.json)

Support & Resources

Documentation

- Getting Started Guide (./GETTING_STARTED.md)
- System Architecture (./ARCHITECTURE.md)
- Automation Guide (./LOMANCO_AUTOMATION_GUIDE.md)

Support Channels

- **Email**: support@ultimate-crm.com
- **Documentation**: https://docs.ultimate-crm.com
- **GitHub Issues**: https://github.com/ultimate-crm/issues

Community

- **Discord**: Join Community (https://discord.gg/ultimate-crm)
- **Forum**: https://forum.ultimate-crm.com
- **Stack Overflow**: Tag `ultimate-crm`

Enterprise-Grade API for Roofing Professionals