ESG Data Management Microservice

API Contract Specification v1.0

Document Information

- **Service Name**: ESG Data Management Microservice
- **API Version**: 1.0.0
- **Document Date**: September 23, 2025
- **Document Owner**: Shireen Hussain J
- **Classification**: Internal Use Only
- **Format**: OpenAPI 3.0.3 Specification

---

1. Service Overview

1.1 Purpose

The ESG Data Management Microservice serves as the core data processing engine for the Sweep platform, responsible for ingesting, validating, calculating, and storing Environmental, Social, and Governance (ESG) metrics across enterprise operations.

1.2 Core Capabilities

- **Data Ingestion**: Batch and streaming data uploads from multiple sources
- **Emissions Calculations**: Scope 1, 2, and 3 emissions using GHG Protocol methodologies
- **Data Validation**: Quality scoring and lineage tracking
- **Metrics Management**: Storage, retrieval, and aggregation of ESG metrics
- **Audit Support**: Complete data lineage and change tracking

1.3 Base URLs

- **Production**: `https://api.sweep.com/v1/esg-data`
- **Staging**: `https://api-staging.sweep.com/v1/esg-data`
- **Development**: `https://api-dev.sweep.com/v1/esg-data`

1.4 Authentication

- **Primary**: OAuth 2.0 Bearer Token (JWT format)
- **Secondary**: API Key authentication
- **Token Expiry**: 24 hours (configurable)

- **Security Headers**: Authorization, X-API-Key, X-Request-ID

---

2. API Endpoints

2.1 Health Check

GET /health

Purpose: Returns service health status and dependency information

Security: No authentication required

Response (200 OK):

```json
{
  "status": "healthy",
  "timestamp": "2025-09-23T15:45:00Z",
  "version": "1.0.0",
  "uptime": "72h 15m 30s",
  "dependencies": {
    "database": {"status": "healthy", "responseTime": "12ms"},
    "cache": {"status": "healthy", "responseTime": "1ms"},
    "messageQueue": {"status": "healthy", "queueDepth": 45}
  }
}
```

2.2 Data Ingestion Endpoints

POST /ingest

Purpose: Upload batch ESG data from various source systems

Headers:

- `Authorization: Bearer <JWT_TOKEN>`
- `Content-Type: application/json`
- `X-Source-System: <SYSTEM_ID>`

Request Body Schema:

- `sourceSystem` (required): Source system identifier
- `ingestionDate` (required): ISO 8601 timestamp

- `dataFormat` (required): JSON, CSV, or XML
- `organizationId` (required): Organization identifier
- `validationLevel`: STRICT, MODERATE, or LENIENT
- `payload` (required): Array of ESG metrics

ESG Metric Properties:

- `entityId` (required): Unique entity identifier
- `metricType` (required): SCOPE_1_EMISSIONS, SCOPE_2_EMISSIONS, SCOPE_3_EMISSIONS, ENERGY_CONSUMPTION, WATER_CONSUMPTION, WASTE_GENERATED, RENEWABLE_ENERGY
- `value` (required): Numeric value (≥ 0)
- `unit` (required): Measurement unit
- `timestamp` (required): ISO 8601 timestamp
- `location`: Geographic information object
- `metadata`: Additional context information

Example Request:

```json
{
  "sourceSystem": "SAP_ERP_PROD",
  "ingestionDate": "2025-09-23T15:30:00Z",
  "dataFormat": "JSON",
  "organizationId": "org-123",
  "validationLevel": "MODERATE",
  "payload": [
    {
      "entityId": "plant-dubai-001",
      "metricType": "SCOPE_1_EMISSIONS",
      "value": 1200.5,
      "unit": "kg CO2e",
      "timestamp": "2025-09-23T12:00:00Z",
      "location": {
        "facilityId": "dubai-manufacturing",
        "latitude": 24.8968,
        "longitude": 55.1632
      }
    }
  ]
}
```

Response (202 Accepted):

```json
{
  "jobId": "job-789xyz-abc123",
  "status": "PROCESSING",
  "recordsReceived": 1250,
  "estimatedProcessingTime": "5-10 minutes",
  "statusUrl": "/ingest/status/job-789xyz-abc123",
  "submittedAt": "2025-09-23T15:30:00Z"
}
```

GET /ingest/status/{jobId}

Purpose: Check the status and progress of a data ingestion job

Path Parameters:

- `jobId`: Ingestion job identifier

Response (200 OK):

```json
{
  "jobId": "job-789xyz-abc123",
  "status": "COMPLETED",
  "progress": {
    "totalRecords": 1250,
    "processedRecords": 1250,
    "validRecords": 1185,
    "invalidRecords": 65,
    "percentComplete": 100
  },
  "results": {
    "recordsIngested": 1185,
    "recordsRejected": 65,
    "calculationsTriggered": 15,
    "dataQualityScore": 94.8
  }
}
```

POST /ingest/stream

Purpose: Real-time data ingestion for IoT sensors and live feeds

Request Body:

```json
{
  "streamId": "sensor-stream-001",
  "entityId": "facility-dubai-001",
  "metricType": "ENERGY_CONSUMPTION",
  "value": 145.7,
  "unit": "kWh",
  "timestamp": "2025-09-23T14:45:30Z",
  "location": {
    "facilityId": "dubai-hq",
    "latitude": 25.2048,
    "longitude": 55.2708
  }
}
```

Response (201 Created):

```json
{
  "id": "metric-stream-001",
  "status": "ACCEPTED",
  "validationWarnings": [],
  "processingTime": "45ms"
}
```

2.3 Metrics Query Endpoints

GET /metrics

Purpose: Retrieve ESG metrics with advanced filtering and pagination

Query Parameters:

- `organizationId` (required): Organization filter
- `metricType`: Filter by metric type
- `entityId`: Filter by entity
- `dateFrom`: Start date (ISO 8601)
- `dateTo`: End date (ISO 8601)
- `page`: Page number (default: 1)
- `pageSize`: Records per page (default: 25, max: 100)
- `sortBy`: Sort field (timestamp, value, metricType)

- `sortOrder`: ASC or DESC (default: DESC)
- `includeCalculated`: Include derived metrics (default: false)

Example URL:

```
GET /metrics?organizationId=org-123&metricType=SCOPE_1_EMISSIONS&dateFrom=2025-01-01T00:00:00Z&page=1&pageSize=50
```

Response (200 OK):

```json
```

```
{
  "pagination": {
    "page": 1,
    "pageSize": 50,
    "totalPages": 24,
    "totalRecords": 1185,
    "hasNext": true,
    "hasPrevious": false
  },
  "metrics": [
    {
      "id": "metric-abc123",
      "entityId": "plant-dubai-001",
      "metricType": "SCOPE_1_EMISSIONS",
      "value": 1200.5,
      "unit": "kg CO2e",
      "timestamp": "2025-09-23T12:00:00Z",
      "location": {
        "facilityId": "dubai-manufacturing",
        "latitude": 24.8968,
        "longitude": 55.1632
      },
      "dataQuality": {
        "score": 96.5,
        "confidence": "HIGH"
      }
    }
  ],
  "aggregations": {
    "totalEmissions": 45678.9,
    "averageDaily": 125.4,
    "trend": "INCREASING"
  }
}
```


GET /metrics/{metricId}

Purpose: Retrieve detailed information about a specific metric

Path Parameters:

- `metricId`: Metric identifier

Response (200 OK):

```
json
```

```
{
  "id": "metric-abc123",
  "entityId": "plant-dubai-001",
  "organizationId": "org-123",
  "metricType": "SCOPE_1_EMISSIONS",
  "value": 1200.5,
  "unit": "kg CO2e",
  "timestamp": "2025-09-23T12:00:00Z",
  "sourceData": {
    "sourceSystem": "SAP_ERP_PROD",
    "sourceRecordId": "ERP-12345",
    "rawValue": 520.0,
    "rawUnit": "liters_diesel"
  },
  "calculationDetails": {
    "emissionFactor": 2.31,
    "emissionFactorSource": "DEFRA_2025",
    "calculationMethod": "ACTIVITY_DATA_BASED",
    "uncertaintyRange": {
      "lower": 1156.5,
      "upper": 1244.5,
      "confidenceLevel": 95
    }
  },
  "audit": {
    "createdAt": "2025-09-23T12:05:00Z",
    "createdBy": "system-integration",
    "version": 2
  }
}
```

2.4 Calculations Endpoints

POST /emissions/calculate

Purpose: Trigger emissions calculations for specified entities and date ranges

Request Body:

```
json
```

```
{
  "organizationId": "org-123",
  "calculationType": "SCOPE_3_CATEGORY_1",
  "entityIds": ["supplier-001", "supplier-002"],
  "dateRange": {
    "startDate": "2025-01-01T00:00:00Z",
    "endDate": "2025-09-23T23:59:59Z"
  },
  "calculationParameters": {
    "methodology": "GHG_PROTOCOL_CORPORATE_VALUE_CHAIN",
    "emissionFactorVersion": "DEFRA_2025",
    "uncertaintyAnalysis": true
  },
  "priority": "HIGH"
}
```

Response (202 Accepted):

json

```
{
  "calculationJobId": "calc-job-456def",
  "status": "QUEUED",
  "estimatedDuration": "15-30 minutes",
  "entitiesInScope": 2,
  "metricsToProcess": 1247,
  "statusUrl": "/emissions/calculate/status/calc-job-456def"
}
```

GET /emissions/calculate/status/{jobId}

Purpose: Check calculation job status and results

Response (200 OK):

json

```
{
  "calculationJobId": "calc-job-456def",
  "status": "COMPLETED",
  "progress": {
    "totalEntities": 2,
    "processedEntities": 2,
    "percentComplete": 100
  },
  "results": {
    "newMetricsCreated": 45,
    "existingMetricsUpdated": 12,
    "totalEmissions": {
      "value": 15678.9,
      "unit": "kg CO2e"
    },
    "categoryBreakdown": [
      {
        "category": "PURCHASED_GOODS_SERVICES",
        "emissions": 8945.2,
        "percentage": 57.1
      }
    ]
  },
  "completedAt": "2025-09-23T15:23:45Z"
}
```

2.5 Data Lineage Endpoints

GET /lineage/{metricId}

Purpose: Retrieve complete data lineage graph for a specific metric

Query Parameters:

- `depth`: Traversal depth (1-5, default: 3)
- `direction`: upstream, downstream, or both (default: both)

Response (200 OK):

```
json
```

```
{
  "rootMetricId": "metric-abc123",
  "lineageGraph": {
    "nodes": [
      {
        "id": "metric-abc123",
        "type": "calculated_metric",
        "metricType": "SCOPE_1_EMISSIONS",
        "value": 1200.5
      },
      {
        "id": "raw-data-001",
        "type": "source_data",
        "sourceSystem": "SAP_ERP_PROD"
      }
    ],
    "edges": [
      {
        "from": "raw-data-001",
        "to": "metric-abc123",
        "relationship": "CALCULATED_FROM"
      }
    ]
  },
  "summary": {
    "totalNodes": 15,
    "sourceDataPoints": 8,
    "calculatedMetrics": 5
  }
}
```

2.6 Audit & Compliance Endpoints

GET /audit/logs

Purpose: Retrieve audit trail logs (requires AUDITOR or ADMIN role)

Query Parameters:

- `entityId`: Filter by entity
- `metricId`: Filter by metric
- `userId`: Filter by user actions
- `action`: Filter by action type

- `dateFrom`: Start date
- `dateTo`: End date
- `page`: Page number
- `pageSize`: Records per page

Response (200 OK):

```json
{
  "pagination": {
    "page": 1,
    "pageSize": 25,
    "totalRecords": 156
  },
  "auditLogs": [
    {
      "id": "audit-log-789",
      "timestamp": "2025-09-23T14:35:22Z",
      "userId": "user-john-doe",
      "userRole": "ESG_ANALYST",
      "action": "METRIC_CREATED",
      "resourceType": "ESG_METRIC",
      "resourceId": "metric-abc123",
      "details": {
        "metricType": "SCOPE_1_EMISSIONS",
        "value": 1200.5
      }
    }
  ]
}
```

---

3. Error Handling

3.1 Standard Error Response Format

```json
```

```
{
  "error": {
    "code": "ERROR_CODE",
    "message": "Human-readable error message",
    "details": "Additional technical details",
    "timestamp": "2025-09-23T15:30:00Z",
    "requestId": "req-123abc-456def",
    "path": "/v1/esg-data/metrics",
    "method": "GET"
  }
}
```

3.2 HTTP Status Codes

Status Code	Error Code	Description
400	INVALID_PAYLOAD	Request body is malformed
400	VALIDATION_FAILED	Field validation errors
400	INVALID_METRIC_TYPE	Unsupported metric type
400	INVALID_UNIT	Unit not compatible with metric
401	UNAUTHORIZED	Missing or invalid token
401	TOKEN_EXPIRED	JWT token expired
403	FORBIDDEN	Insufficient permissions
404	METRIC_NOT_FOUND	Specified metric doesn't exist
409	DUPLICATE_METRIC	Metric already exists
422	CALCULATION_ERROR	Error in calculation process
429	RATE_LIMIT_EXCEEDED	Too many requests
500	INTERNAL_ERROR	Unexpected server failure
503	SERVICE_UNAVAILABLE	Service temporarily unavailable

---

4. Authentication & Authorization

4.1 OAuth 2.0 Flow

bash

```
# Step 1: Get Access Token
curl -X POST https://auth.sweep.com/oauth/token \
  -H "Content-Type: application/x-www-form-urlencoded" \
  -d "grant_type=client_credentials&client_id=YOUR_CLIENT_ID&client_secret=YOUR_SECRET"

# Response
{
  "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "token_type": "Bearer",
  "expires_in": 86400
}

# Step 2: Use Token
curl -X GET "https://api.sweep.com/v1/esg-data/metrics?organizationId=org-123" \
  -H "Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."
```

4.2 Role-Based Access Control

Role	Data Ingestion	Metrics Read	Calculations	Audit Logs	Admin Operations
ESG Analyst	✓	✓	✓	✗	✗
Sustainability Manager	✓	✓	✓	✓	✗
System Administrator	✓	✓	✓	✓	✓
Auditor	✗	✓	✗	✓	✗
Supplier	✓ (Limited)	✓ (Own Data)	✗	✗	✗

---

5. Rate Limiting

5.1 Rate Limits by Client Type

Client Type	Requests/Hour	Burst Limit
Web UI	10,000	100/min
API Client	5,000	50/min
Supplier Portal	1,000	20/min
Integration Service	50,000	500/min

5.2 Rate Limit Headers

```
http
```

```
X-RateLimit-Limit: 1000
X-RateLimit-Remaining: 999
X-RateLimit-Reset: 2025-09-23T17:00:00Z
```

---

6. Data Validation Rules

6.1 Metric Value Validation

- **Numeric Values**: Must be non-negative numbers
- **Units**: Must match predefined unit dictionary per metric type
- **Timestamps**: Valid ISO 8601 format, not future-dated
- **Location**: Latitude (-90 to 90), Longitude (-180 to 180)

6.2 Business Logic Validation

- **Emission Factors**: Must be available for calculation date
- **Organizational Scope**: Entity must belong to requesting organization
- **Data Consistency**: Related metrics must be logically consistent

---

7. Performance & SLA

7.1 Response Time Targets

Endpoint Category	Target Response Time	SLA Percentile
Single Metric GET	< 200ms	95th percentile
Metrics Query	< 1s	95th percentile
Data Ingestion POST	< 5s	99th percentile
Calculation Triggers	< 2s	95th percentile

7.2 Availability & Reliability

- **Uptime SLA**: 99.9% (8.77 hours downtime/year)
- **Error Rate**: < 0.1% for successful requests
- **Data Consistency**: 99.99% accuracy for calculations

7.3 Scalability Limits

- **Max Payload Size**: 50MB per request
- **Max Records per Batch**: 10,000 records

- **Concurrent Requests**: 1,000 per organization
- **Data Retention**: 7 years active, 20 years archived

---

8. Integration Examples

8.1 JavaScript Example

javascript

```javascript
// Complete data ingestion flow
class ESGDataClient {
  constructor(accessToken) {
    this.baseUrl = 'https://api.sweep.com/v1/esg-data';
    this.headers = {
      'Authorization': `Bearer ${accessToken}`,
      'Content-Type': 'application/json'
    };
  }

  async ingestData(payload) {
    const response = await fetch(`${this.baseUrl}/ingest`, {
      method: 'POST',
      headers: this.headers,
      body: JSON.stringify({
        sourceSystem: 'CUSTOM_INTEGRATION',
        ingestionDate: new Date().toISOString(),
        dataFormat: 'JSON',
        organizationId: 'org-123',
        payload: payload
      })
    });
    return response.json();
  }

  async queryMetrics(filters) {
    const params = new URLSearchParams(filters);
    const response = await fetch(`${this.baseUrl}/metrics?${params}`, {
      headers: this.headers
    });
    return response.json();
  }
}

// Usage
const client = new ESGDataClient('your-access-token');
const result = await client.ingestData([
  {
    entityId: 'facility-001',
    metricType: 'SCOPE_1_EMISSIONS',
    value: 1200.5,
    unit: 'kg CO2e',
    timestamp: '2025-09-23T12:00:00Z'
  }
]);
```

8.2 Python Example

```python
import requests
from datetime import datetime

class ESGDataClient:
    def __init__(self, access_token):
        self.base_url = 'https://api.sweep.com/v1/esg-data'
        self.headers = {
            'Authorization': f'Bearer {access_token}',
            'Content-Type': 'application/json'
        }

    def ingest_data(self, payload):
        data = {
            'sourceSystem': 'PYTHON_CLIENT',
            'ingestionDate': datetime.utcnow().isoformat() + 'Z',
            'dataFormat': 'JSON',
            'organizationId': 'org-123',
            'payload': payload
        }
        response = requests.post(f'{self.base_url}/ingest',
                                 json=data, headers=self.headers)
        return response.json()

    def query_metrics(self, **filters):
        response = requests.get(f'{self.base_url}/metrics',
                                params=filters, headers=self.headers)
        return response.json()

# Usage
client = ESGDataClient('your-access-token')
result = client.ingest_data([{
    'entityId': 'facility-001',
    'metricType': 'ENERGY_CONSUMPTION',
    'value': 15600.0,
    'unit': 'kWh',
    'timestamp': '2025-09-23T08:00:00Z'
}])
```

---

9. Monitoring & Observability

9.1 Metrics Exposed

The service exposes Prometheus metrics at `/metrics` endpoint:

- `esg_data_requests_total`: Total API requests by endpoint and status
- `esg_data_response_time`: Response time histogram
- `esg_data_ingestion_records_total`: Total records ingested
- `esg_data_calculation_jobs_total`: Calculation jobs by status
- `esg_data_db_connections`: Database connection pool metrics

9.2 Structured Logging

All logs follow JSON format:

```json
{
  "timestamp": "2025-09-23T15:45:00Z",
  "level": "INFO",
  "service": "esg-data-management",
  "requestId": "req-123abc-456def",
  "userId": "user-john-doe",
  "organizationId": "org-123",
  "action": "METRIC_QUERY",
  "duration": 156,
  "status": "SUCCESS"
}
```

---

10. Security & Compliance

10.1 Data Protection

- **Encryption at Rest**: AES-256 for database and file storage
- **Encryption in Transit**: TLS 1.3 for all API communications
- **Data Retention**: Configurable policies per organization
- **Right to Deletion**: API endpoints for data removal requests

10.2 Compliance Standards

- **SOC 2 Type II**: Security, availability, and confidentiality
- **ISO 27001**: Information security management
- **GDPR**: EU data protection regulations
- **CCPA**: California Consumer Privacy Act

10.3 Audit Controls

- **Comprehensive Audit Trails**: All data operations logged
- **Real-time Security Monitoring**: SIEM integration
- **Access Controls**: Role-based permissions with principle of least privilege
- **Penetration Testing**: Quarterly security assessments

---

11. Support & Resources

11.1 Developer Resources

- **Interactive Documentation**: https://api.sweep.com/docs
- **Postman Collection**: https://docs.sweep.com/api/postman/esg-data-v1.json
- **SDK Libraries**: JavaScript, Python, Java available
- **Code Examples**: https://github.com/sweep-platform/api-examples

11.2 Support Channels

- **Documentation**: https://docs.sweep.com/api/esg-data-management
- **Support Portal**: https://support.sweep.com
- **Developer Slack**: #api-support channel
- **Status Page**: https://status.sweep.com

11.3 SLA & Response Times

Support Tier	Response Time	Availability	Support Hours
Enterprise	1 hour	99.9%	24/7
Professional	4 hours	99.5%	Business hours
Standard	24 hours	99.0%	Business hours

---

12. Version History & Changelog

12.1 Current Version

- **Version**: 1.0.0
- **Release Date**: September 23, 2025
- **Status**: Current

12.2 Deprecation Policy

- **Minor Versions**: No breaking changes, new features added
- **Major Versions**: Breaking changes allowed with 6-month migration period
- **Support Timeline**: Minimum 12 months for major version retirement

---

Document Control

- **Created By**: Shireen Hussain J
- **Technical Review**: Engineering Lead
- **Security Review**: Security Team Lead
- **Last Updated**: September 23, 2025
- **Next Review**: December 23, 2025
- **Classification**: Confidential - Internal Use Only