Compliance & Reporting Service

API Contract Specification v1.0

Document Information

- **Service Name:** Compliance & Reporting Service
- **API Version:** 1.0.0
- **Document Date:** September 25, 2025
- **Document Owner:** Shireen Hussain J
- **Classification:** Internal Use Only
- **Technology Stack:** Java 17 + Spring Boot + Apache POI

1. Service Overview

1.1 Purpose

The Compliance & Reporting Service manages the generation, validation, and submission of ESG reports across multiple regulatory standards. It serves as the central engine for transforming raw ESG data into compliant, audit-ready reports for various stakeholders and regulatory bodies.

1.2 Core Capabilities

- **Multi-Standard Reporting:** CSRD, SFDR, ISSB, GRI, CDP, GHG Protocol compliance
- **Template Management:** Dynamic report templates with regulatory updates
- **Report Generation:** Automated creation of PDF, Word, Excel, and XML formats
- **Approval Workflows:** Multi-level review and approval processes
- **Submission Management:** Direct integration with regulatory portals
- **Audit Trail:** Complete documentation and version control
- **Validation Engine:** Compliance rule checking and gap analysis

1.3 Base URLs

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

1.4 Authentication

- **Primary:** OAuth 2.0 Bearer Token (JWT format)
- **Service-to-Service:** API Key authentication

- **Required Roles:** ESG_ANALYST, SUSTAINABILITY_MANAGER, COMPLIANCE_OFFICER, AUDITOR

2. API Endpoints

2.1 Health Check

GET /health

- **Purpose:** Returns service health and regulatory framework status

Response (200 OK):

```
json
```


```
{
  "status": "healthy",
  "timestamp": "2025-09-25T15:45:00Z",
  "version": "1.0.0",
  "uptime": "120h 45m 30s",
  "regulatoryFrameworks": {
    "CSRD": {
      "status": "active",
      "version": "2024.1",
      "lastUpdate": "2025-09-01T00:00:00Z",
      "templatesLoaded": 12
    },
    "SFDR": {
      "status": "active",
      "version": "2024.2",
      "paiIndicators": 18,
      "templatesLoaded": 8
    },
    "GRI": {
      "status": "active",
      "version": "2023",
      "universalStandards": 3,
      "topicStandards": 33
    }
  },
  "reportQueue": {
    "pendingReports": 15,
    "inProgress": 8,
    "averageProcessingTime": "12m 30s"
  }
}
```

2.2 Report Generation

POST /reports/generate

- **Purpose:** Generate compliance reports based on specified standards and parameters

Request Body:

json

```
{
  "reportJobId": "report-job-abc123",
  "organizationId": "org-123",
  "reportConfig": {
    "standard": "CSRD",
    "template": "ESRS_E1_CLIMATE_CHANGE",
    "reportingPeriod": {
      "startDate": "2024-01-01T00:00:00Z",
      "endDate": "2024-12-31T23:59:59Z",
      "fiscalYear": "2024"
    },
    "scope": {
      "entities": ["entity-dubai-hq", "entity-abu-dhabi-plant"],
      "geographies": ["UAE", "KSA"],
      "businessUnits": ["manufacturing", "operations"]
    },
    "outputFormats": ["PDF", "XLSX", "XML"],
    "language": "EN",
    "currency": "USD"
  },
  "dataFilters": {
    "includeEstimatedData": true,
    "qualityThreshold": 85.0,
    "includeThirdPartyData": true,
    "dataAssurance": "LIMITED"
  },
  "customizations": {
    "includeBranding": true,
    "executiveSummary": true,
    "detailedBreakdowns": true,
    "comparativePeriods": ["2023", "2022"]
  }
}
```

Response (202 Accepted):

json

```
{
  "reportJobId": "report-job-abc123",
  "status": "PROCESSING",
  "standard": "CSRD",
  "template": "ESRS_E1_CLIMATE_CHANGE",
  "estimatedCompletionTime": "15-25 minutes",
  "dataPointsToProcess": 1247,
  "calculationsRequired": 45,
  "statusUrl": "/reports/generate/status/report-job-abc123",
  "submittedAt": "2025-09-25T16:00:00Z"
}
```


GET /reports/generate/status/{jobId}

- **Purpose:** Check report generation status and retrieve completed reports

Response (200 OK) - Processing:

json


```
{
  "reportJobId": "report-job-abc123",
  "status": "PROCESSING",
  "progress": {
    "dataCollection": "COMPLETED",
    "calculations": "IN_PROGRESS",
    "validation": "PENDING",
    "formatting": "PENDING",
    "percentComplete": 45
  },
  "estimatedCompletionTime": "12 minutes"
}
```


Response (200 OK) - Completed:

json

```
{
  "reportJobId": "report-job-abc123",
  "status": "COMPLETED",
  "progress": {
    "dataCollection": "COMPLETED",
    "calculations": "COMPLETED",
    "validation": "COMPLETED",
    "formatting": "COMPLETED",
    "percentComplete": 100
  },
  "results": {
    "reportsGenerated": 3,
    "totalPages": 156,
    "dataPointsProcessed": 1247,
    "calculationsPerformed": 45,
    "validationIssues": 2,
    "outputs": [
      {
        "format": "PDF",
        "fileSize": "12.4MB",
        "downloadUrl": "/reports/download/report-job-abc123.pdf",
        "expiresAt": "2025-09-31T16:00:00Z"
      },
      {
        "format": "XLSX",
        "fileSize": "3.8MB",
        "downloadUrl": "/reports/download/report-job-abc123.xlsx",
        "worksheets": 12
      }
    ]
  },
  "compliance": {
    "standardCompliance": "FULLY_COMPLIANT",
    "mandatoryDisclosures": {
      "total": 85,
      "completed": 83,
      "missing": 2,
      "completenessPercentage": 97.6
    },
    "validationResults": [
      {
        "ruleId": "ESRS-E1-6",
        "status": "WARNING",
        "message": "Energy consumption data missing for Q4 renewable sources",
        "impact": "LOW",
        "recommendation": "Include estimated renewable energy consumption"
```

```
      }
    ]
  },
  "completedAt": "2025-09-25T16:18:45Z"
}
```

2.3 Report Templates Management

GET /templates

- **Purpose:** List available report templates by standard

Query Parameters:

- `standard`: Filter by regulatory standard (CSRD, SFDR, ISSB, GRI, CDP, GHG_PROTOCOL)
- `category`: Filter by category (ENVIRONMENTAL, SOCIAL, GOVERNANCE)
- `language`: Template language (EN, FR, DE, ES)
- `version`: Template version

Response (200 OK):

```json
```

```
{
  "templates": [
    {
      "templateId": "CSRD_ESRS_E1",
      "name": "CSRD - Climate Change (ESRS E1)",
      "standard": "CSRD",
      "category": "ENVIRONMENTAL",
      "version": "1.2.0",
      "description": "CSRD compliance template for ESRS E1 Climate Change disclosure",
      "languages": ["EN", "FR", "DE"],
      "requiredDataPoints": [
        "scope_1_emissions",
        "scope_2_emissions",
        "scope_3_emissions",
        "energy_consumption",
        "renewable_energy_percentage"
      ],
      "mandatoryDisclosures": 15,
      "optionalDisclosures": 8,
      "estimatedPages": 25,
      "lastUpdated": "2025-08-15T10:00:00Z",
      "regulatoryDeadline": "2026-03-31T00:00:00Z"
    }
  ],
  "totalTemplates": 45,
  "standardsSupported": 6,
  "lastFrameworkUpdate": "2025-09-01T00:00:00Z"
}
```


GET /templates/{templateId}

- **Purpose:** Get detailed template information and requirements

Response (200 OK):

```
json
```

```
{
  "templateId": "CSRD_ESRS_E1",
  "name": "CSRD - Climate Change (ESRS E1)",
  "standard": "CSRD",
  "version": "1.2.0",
  "structure": {
    "sections": [
      {
        "sectionId": "E1-1",
        "title": "Transition plan for climate change mitigation",
        "mandatory": true,
        "subsections": [
          {
            "subsectionId": "E1-1.1",
            "title": "Transition plan",
            "dataRequirements": [
              "decarbonization_targets",
              "transition_timeline",
              "investment_plans"
            ],
            "validationRules": [
              {
                "ruleId": "E1-1.1-R1",
                "description": "Transition plan must include specific targets",
                "severity": "MANDATORY"
              }
            ]
          }
        ]
      }
    ]
  },
  "dataMapping": {
    "scope_1_emissions": {
      "metricTypes": ["SCOPE_1_EMISSIONS"],
      "aggregation": "SUM",
      "units": ["kg CO2e", "tonnes CO2e"],
      "timeGranularity": "ANNUAL"
    }
  },
  "outputOptions": {
    "formats": ["PDF", "DOCX", "XLSX", "XML"],
    "customBranding": true,
    "multiLanguage": true,
    "digitalSignature": true
```

```
  }
}
```

2.4 Compliance Validation

POST /validate/compliance

- **Purpose:** Validate data completeness and compliance before report generation

Request Body:

json

```
{
  "validationJobId": "validation-job-456def",
  "organizationId": "org-123",
  "validationConfig": {
    "standard": "CSRD",
    "template": "ESRS_E1_CLIMATE_CHANGE",
    "reportingPeriod": {
      "startDate": "2024-01-01T00:00:00Z",
      "endDate": "2024-12-31T23:59:59Z"
    },
    "strictMode": true,
    "includeWarnings": true
  },
  "scope": {
    "entities": ["entity-dubai-hq", "entity-abu-dhabi-plant"],
    "dataTypes": ["EMISSIONS", "ENERGY", "WASTE", "WATER"]
  }
}
```

Response (200 OK):

json

```
{
  "validationJobId": "validation-job-456def",
  "status": "COMPLETED",
  "overallCompliance": {
    "status": "PARTIALLY_COMPLIANT",
    "score": 87.3,
    "readinessLevel": "NEEDS_IMPROVEMENT"
  },
  "mandatoryRequirements": {
    "total": 15,
    "satisfied": 13,
    "missing": 2,
    "complianceRate": 86.7
  },
  "validationResults": [
    {
      "requirementId": "ESRS-E1-1",
      "title": "Transition plan for climate change mitigation",
      "status": "COMPLIANT",
      "completeness": 100,
      "dataQuality": 95.2
    },
    {
      "requirementId": "ESRS-E1-5",
      "title": "Energy consumption and mix",
      "status": "NON_COMPLIANT",
      "completeness": 65,
      "issues": [
        {
          "type": "MISSING_DATA",
          "severity": "CRITICAL",
          "description": "Renewable energy percentage missing for Q3 and Q4",
          "dataPoints": ["renewable_energy_q3", "renewable_energy_q4"],
          "recommendation": "Collect or estimate missing renewable energy data"
        }
      ]
    }
  ],
  "gapAnalysis": {
    "criticalGaps": 2,
    "warnings": 5,
    "recommendations": [
      "Implement automated renewable energy tracking",
      "Establish quarterly data validation procedures",
      "Consider third-party data assurance for material metrics"
    ],
```

```
    "estimatedEffortToCompliance": "40-60 hours"
  },
  "completedAt": "2025-09-25T16:25:00Z"
}
```

2.5 Approval Workflows

POST /approvals/initiate

- **Purpose:** Start approval workflow for generated reports

Request Body:

```
json
```

```
{
  "approvalJobId": "approval-job-789ghi",
  "reportJobId": "report-job-abc123",
  "organizationId": "org-123",
  "workflowConfig": {
    "workflowType": "CSRD_APPROVAL",
    "approvers": [
      {
        "level": 1,
        "role": "ESG_ANALYST",
        "users": ["user-sarah-chen"],
        "required": true,
        "deadline": "2025-09-26T17:00:00Z"
      },
      {
        "level": 2,
        "role": "SUSTAINABILITY_MANAGER",
        "users": ["user-michael-brown"],
        "required": true,
        "deadline": "2025-09-26T17:00:00Z"
      },
      {
        "level": 3,
        "role": "COMPLIANCE_OFFICER",
        "users": ["user-lisa-wilson", "user-david-kumar"],
        "required": false,
        "parallelApproval": true
      }
    ],
    "escalationRules": {
      "autoEscalateAfter": "24h",
      "escalateTo": "user-ceo-office"
    }
  },
  "notifications": {
    "email": true,
    "inApp": true,
    "webhook": "https://client.sweep.com/webhooks/approval-status"
  }
}
```

Response (202 Accepted):

```
json
```

```
{
  "approvalJobId": "approval-job-789ghi",
  "workflowId": "workflow-123abc",
  "status": "INITIATED",
  "currentLevel": 1,
  "currentApprovers": ["user-sarah-chen"],
  "overallDeadline": "2025-09-26T17:00:00Z",
  "statusUrl": "/approvals/status/approval-job-789ghi",
  "initiatedAt": "2025-09-25T16:30:00Z"
}
```


POST /approvals/{approvalJobId}/action

- **Purpose:** Approve, reject, or request changes to reports

Request Body:

json


```
{
  "action": "APPROVE",
  "approver": "user-sarah-chen",
  "level": 1,
  "comments": "Report looks comprehensive and meets CSRD requirements",
  "conditions": [],
  "attachments": [
    {
      "name": "approval_certificate.pdf",
      "url": "/attachments/approval_certificate_789ghi.pdf"
    }
  ]
}
```


Response (200 OK):

json

```
{
  "approvalJobId": "approval-job-789ghi",
  "actionResult": "APPROVED_LEVEL_1",
  "nextLevel": 2,
  "nextApprovers": ["user-michael-brown"],
  "workflowStatus": "IN_PROGRESS",
  "actionRecorded": "2025-09-25T17:15:00Z",
  "notifications": {
    "sent": ["user-michael-brown"],
    "method": ["EMAIL", "IN_APP"]
  }
}
```


2.6 Regulatory Submission

POST /submit/regulatory

- **Purpose:** Submit approved reports to regulatory portals

Request Body:

```
json
```

```
{
  "submissionJobId": "submission-job-abc456",
  "reportJobId": "report-job-abc123",
  "organizationId": "org-123",
  "submissionConfig": {
    "standard": "CSRD",
    "regulatoryPortal": "ESMA_SINGLE_ACCESS_POINT",
    "submissionType": "ANNUAL_SUSTAINABILITY_STATEMENT",
    "deadlineDate": "2025-04-30T00:00:00Z",
    "testMode": false
  },
  "authorizedBy": {
    "userId": "user-cfo-john-smith",
    "role": "CHIEF_FINANCIAL_OFFICER",
    "digitalSignature": "base64_encoded_signature",
    "timestamp": "2025-09-25T18:00:00Z"
  },
  "submissionMetadata": {
    "lei": "LEI1234567890ABCDEF",
    "fiscalYear": "2024",
    "preparationDate": "2025-09-25T00:00:00Z",
    "assuranceLevel": "LIMITED",
    "assuranceProvider": "PwC Dubai"
  }
}
```


Response (202 Accepted):


```
json

{
  "submissionJobId": "submission-job-abc456",
  "status": "PROCESSING",
  "regulatoryPortal": "ESMA_SINGLE_ACCESS_POINT",
  "submissionReference": "ESMA-2025-ORG123-CSRD-001",
  "estimatedProcessingTime": "5-10 minutes",
  "validationSteps": [
    "DIGITAL_SIGNATURE_VERIFICATION",
    "SCHEMA_VALIDATION",
    "BUSINESS_RULES_CHECK",
    "PORTAL_SUBMISSION"
  ],
  "statusUrl": "/submit/regulatory/status/submission-job-abc456"
}
```


GET /submit/regulatory/status/{jobId}

- **Purpose:** Check regulatory submission status

Response (200 OK):

```json
{
  "submissionJobId": "submission-job-abc456",
  "status": "COMPLETED",
  "submissionResults": {
    "submissionSuccess": true,
    "regulatoryReference": "ESMA-2025-ORG123-CSRD-001",
    "submissionTimestamp": "2025-09-25T18:12:45Z",
    "acknowledgmentReceipt": {
      "receiptId": "RCP-ESMA-456789",
      "downloadUrl": "/submissions/receipts/RCP-ESMA-456789.pdf"
    }
  },
  "validationSteps": [
    {
      "step": "DIGITAL_SIGNATURE_VERIFICATION",
      "status": "PASSED",
      "completedAt": "2025-09-25T18:05:15Z"
    },
    {
      "step": "SCHEMA_VALIDATION",
      "status": "PASSED",
      "completedAt": "2025-09-25T18:08:30Z"
    },
    {
      "step": "PORTAL_SUBMISSION",
      "status": "PASSED",
      "completedAt": "2025-09-25T18:12:45Z"
    }
  ],
  "compliance": {
    "deadlineMet": true,
    "daysBeforeDeadline": 218,
    "submissionComplete": true
  }
}
```

2.7 Audit Trail & Documentation

GET /audit/reports

- **Purpose:** Retrieve audit trail for report generation and approval activities

Query Parameters:

- `organizationId` (required): Organization filter
- `reportJobId`: Specific report job
- `standard`: Filter by regulatory standard
- `dateFrom`: Start date filter
- `dateTo`: End date filter
- `action`: Filter by action type
- `userId`: Filter by user actions

Response (200 OK):

```json
```

```
{
  "auditTrail": [
    {
      "auditId": "audit-001",
      "timestamp": "2025-09-25T16:00:00Z",
      "action": "REPORT_GENERATION_INITIATED",
      "userId": "user-sarah-chen",
      "userRole": "ESG_ANALYST",
      "resourceType": "COMPLIANCE_REPORT",
      "resourceId": "report-job-abc123",
      "details": {
        "standard": "CSRD",
        "template": "ESRS_E1_CLIMATE_CHANGE",
        "reportingPeriod": "2024"
      },
      "ipAddress": "10.0.1.45",
      "sessionId": "session-789def"
    },
    {
      "auditId": "audit-002",
      "timestamp": "2025-09-25T17:15:00Z",
      "action": "REPORT_APPROVED",
      "userId": "user-sarah-chen",
      "userRole": "ESG_ANALYST",
      "resourceType": "APPROVAL_WORKFLOW",
      "resourceId": "approval-job-789ghi",
      "details": {
        "approvalLevel": 1,
        "comments": "Report meets CSRD requirements",
        "nextApprover": "user-michael-brown"
      }
    }
  ],
  "pagination": {
    "page": 1,
    "pageSize": 25,
    "totalRecords": 156,
    "totalPages": 7
  }
}
```


2.8 Report Download & File Management

GET /reports/download/{filename}

- **Purpose:** Download generated report files

Response (200 OK):

- Content-Type: application/pdf, application/vnd.openxmlformats-officedocument.spreadsheetml.sheet, etc.
- Content-Disposition: attachment; filename="CSRD_Climate_Report_2024.pdf"

GET /reports/{reportJobId}/metadata

- **Purpose:** Get report metadata and file information

Response (200 OK):

json


```
{
  "reportJobId": "report-job-abc123",
  "generatedFiles": [
    {
      "fileId": "file-001",
      "format": "PDF",
      "filename": "CSRD_Climate_Report_2024.pdf",
      "size": "12.4MB",
      "checksum": "sha256:abc123...",
      "createdAt": "2025-09-24T16:18:45Z",
      "downloadUrl": "/reports/download/report-job-abc123.pdf",
      "expiresAt": "2025-10-24T16:18:45Z"
    }
  ],
  "retention": {
    "retentionPeriod": "7 years",
    "archiveDate": "2032-09-25T00:00:00Z"
  }
}
```


3. Error Handling

3.1 Standard Error Response Format

json

```
{
  "error": {
    "code": "COMPLIANCE_VALIDATION_FAILED",
    "message": "Report validation failed due to missing mandatory disclosures",
    "details": "ESRS E1-5 energy consumption data incomplete",
    "timestamp": "2025-09-25T16:30:00Z",
    "requestId": "req-compliance-123abc",
    "path": "/v1/compliance-reporting/validate/compliance",
    "method": "POST",
    "complianceInfo": {
      "standard": "CSRD",
      "template": "ESRS_E1_CLIMATE_CHANGE",
      "missingRequirements": 2,
      "complianceScore": 87.3
    }
  }
}
```


3.2 Compliance-Specific Error Codes

Status Code	Error Code	Description
400	INVALID_STANDARD	Regulatory standard not supported
400	INVALID_TEMPLATE	Report template not found or outdated
400	INVALID_REPORTING_PERIOD	Reporting period format or range invalid
422	COMPLIANCE_VALIDATION_FAILED	Data doesn't meet regulatory requirements
422	INSUFFICIENT_DATA	Not enough data for report generation
422	APPROVAL_WORKFLOW_ERROR	Error in approval process execution
422	REGULATORY_SUBMISSION_FAILED	Submission to regulatory portal failed
424	TEMPLATE_OUTDATED	Template version no longer compliant
503	REGULATORY_PORTAL_UNAVAILABLE	External regulatory system unavailable

4. Performance & SLA

4.1 Processing Time Targets

Operation Type	Target Processing Time	SLA Percentile
Report Generation	< 20 minutes	90th percentile
Compliance Validation	< 2 minutes	95th percentile
Template Retrieval	< 5 seconds	95th percentile
Approval Actions	< 10 seconds	99th percentile
Regulatory Submission	< 10 minutes	90th percentile

4.2 Report Quality Standards

- **Template Accuracy:** 100% compliance with latest regulatory versions
- **Data Completeness:** ≥ 95% for mandatory disclosures
- **Validation Accuracy:** ≥ 99.5% correct identification of compliance gaps
- **Format Consistency:** 100% adherence to regulatory formatting requirements

4.3 Scalability Limits

- **Concurrent Report Generation:** 50 per organization
- **Max Report Size:** 500MB per output file
- **Data Points per Report:** Up to 50,000 metrics
- **Approval Workflow Participants:** Up to 20 approvers per workflow

5. Integration Examples

5.1 Java Spring Boot Client

java

```java
@Service
public class ComplianceReportingClient {
    private final WebClient webClient;
    private final String baseUrl = "https://api.sweep.com/v1/compliance-reporting";

    public ComplianceReportingClient(WebClient.Builder webClientBuilder) {
        this.webClient = webClientBuilder
            .baseUrl(baseUrl)
            .build();
    }

    public Mono<ReportGenerationResponse> generateCSRDReport(
            String organizationId,
            String template,
            LocalDate startDate,
            LocalDate endDate) {

        ReportGenerationRequest request = ReportGenerationRequest.builder()
            .reportJobId("report-job-" + UUID.randomUUID())
            .organizationId(organizationId)
            .reportConfig(ReportConfig.builder()
                .standard("CSRD")
                .template(template)
                .reportingPeriod(ReportingPeriod.builder()
                    .startDate(startDate.atStartOfDay().atZone(ZoneOffset.UTC))
                    .endDate(endDate.atTime(LocalTime.MAX).atZone(ZoneOffset.UTC))
                    .build())
                .outputFormats(List.of("PDF", "XLSX"))
                .build())
            .build();

        return webClient.post()
            .uri("/reports/generate")
            .header("Authorization", "Bearer " + getAccessToken())
            .bodyValue(request)
            .retrieve()
            .bodyToMono(ReportGenerationResponse.class);
    }

    public Mono<ComplianceValidationResponse> validateCompliance(
            String organizationId,
            String standard,
            String template,
            LocalDate startDate,
            LocalDate endDate) {
```

```
        ComplianceValidationRequest request = ComplianceValidationRequest.builder()
            .validationJobId("validation-job-" + UUID.randomUUID())
            .organizationId(organizationId)
            .validationConfig(ValidationConfig.builder()
                .standard(standard)
                .template(template)
                .strictMode(true)
                .includeWarnings(true)
                .build())
            .build();

        return webClient.post()
            .uri("/validate/compliance")
            .header("Authorization", "Bearer " + getAccessToken())
            .bodyValue(request)
            .retrieve()
            .bodyToMono(ComplianceValidationResponse.class);
    }

    public Mono<ReportStatus> pollReportStatus(String reportJobId) {
        return webClient.get()
            .uri("/reports/generate/status/{jobId}", reportJobId)
            .header("Authorization", "Bearer " + getAccessToken())
            .retrieve()
            .bodyToMono(ReportStatus.class);
    }

    private String getAccessToken() {
        // Implementation for OAuth token retrieval
        return tokenService.getValidAccessToken();
    }
}

// Usage Example
@RestController
public class ReportController {
    @Autowired
    private ComplianceReportingClient complianceClient;

    @PostMapping("/generate-csrd-report")
    public Mono<ResponseEntity<ReportGenerationResponse>> generateReport(
            @RequestBody ReportRequest request) {

        return complianceClient.generateCSRDReport(
                request.getOrganizationId(),
                "ESRS_E1_CLIMATE_CHANGE",
                request.getStartDate(),
```

```
                request.getEndDate())
            .map(ResponseEntity::ok);
    }

    @GetMapping("/report-status/{jobId}")
    public Mono<ResponseEntity<ReportStatus>> getReportStatus(
            @PathVariable String jobId) {

        return complianceClient.pollReportStatus(jobId)
            .map(ResponseEntity::ok);
    }
}
```

5.2 Python Integration Example

python

```
import requests
import time
from datetime import datetime, timedelta
from typing import Dict, List, Optional

class ComplianceReportingClient:
    def __init__(self, access_token: str, base_url: str = "https://api.sweep.com/v1/compliance-reporting"):
        self.base_url = base_url
        self.headers = {
            'Authorization': f'Bearer {access_token}',
            'Content-Type': 'application/json'
        }

    def generate_report(self,
                        organization_id: str,
                        standard: str,
                        template: str,
                        start_date: datetime,
                        end_date: datetime,
                        output_formats: List[str] = None) -> Dict:
        """Generate compliance report"""
        if output_formats is None:
            output_formats = ["PDF", "XLSX"]

        payload = {
            'reportJobId': f'report-job-{int(time.time())}',
            'organizationId': organization_id,
            'reportConfig': {
                'standard': standard,
                'template': template,
                'reportingPeriod': {
                    'startDate': start_date.isoformat() + 'Z',
                    'endDate': end_date.isoformat() + 'Z',
                    'fiscalYear': str(start_date.year)
                },
                'outputFormats': output_formats,
                'language': 'EN',
                'currency': 'USD'
            },
            'dataFilters': {
                'includeEstimatedData': True,
                'qualityThreshold': 85.0,
                'includeThirdPartyData': True
            }
        }
```

```
    response = requests.post(
        f'{self.base_url}/reports/generate',
        json=payload,
        headers=self.headers
    )
    response.raise_for_status()
    return response.json()

def get_report_status(self, report_job_id: str) -> Dict:
    """Check report generation status"""
    response = requests.get(
        f'{self.base_url}/reports/generate/status/{report_job_id}',
        headers=self.headers
    )
    response.raise_for_status()
    return response.json()

def validate_compliance(self,
                        organization_id: str,
                        standard: str,
                        template: str,
                        start_date: datetime,
                        end_date: datetime) -> Dict:
    """Validate compliance before report generation"""
    payload = {
        'validationJobId': f'validation-job-{int(time.time())}',
        'organizationId': organization_id,
        'validationConfig': {
            'standard': standard,
            'template': template,
            'reportingPeriod': {
                'startDate': start_date.isoformat() + 'Z',
                'endDate': end_date.isoformat() + 'Z'
            },
            'strictMode': True,
            'includeWarnings': True
        },
        'scope': {
            'entities': ['all'],
            'dataTypes': ['EMISSIONS', 'ENERGY', 'WASTE', 'WATER']
        }
    }

    response = requests.post(
        f'{self.base_url}/validate/compliance',
        json=payload,
        headers=self.headers
```

```
    )
    response.raise_for_status()
    return response.json()

def initiate_approval_workflow(self,
                               report_job_id: str,
                               organization_id: str,
                               workflow_type: str = "CSRD_APPROVAL") -> Dict:
    """Start approval workflow for generated report"""
    payload = {
        'approvalJobId': f'approval-job-{int(time.time())}',
        'reportJobId': report_job_id,
        'organizationId': organization_id,
        'workflowConfig': {
            'workflowType': workflow_type,
            'approvers': [
                {
                    'level': 1,
                    'role': 'ESG_ANALYST',
                    'required': True,
                    'deadline': (datetime.now() + timedelta(days=2)).isoformat() + 'Z'
                },
                {
                    'level': 2,
                    'role': 'SUSTAINABILITY_MANAGER',
                    'required': True,
                    'deadline': (datetime.now() + timedelta(days=4)).isoformat() + 'Z'
                }
            ]
        }
    }

    response = requests.post(
        f'{self.base_url}/approvals/initiate',
        json=payload,
        headers=self.headers
    )
    response.raise_for_status()
    return response.json()

def submit_to_regulatory_portal(self,
                                report_job_id: str,
                                organization_id: str,
                                lei: str,
                                authorized_by_user_id: str) -> Dict:
    """Submit approved report to regulatory portal"""
    payload = {
```

```
            'submissionJobId': f'submission-job-{int(time.time())}',
            'reportJobId': report_job_id,
            'organizationId': organization_id,
            'submissionConfig': {
                'standard': 'CSRD',
                'regulatoryPortal': 'ESMA_SINGLE_ACCESS_POINT',
                'submissionType': 'ANNUAL_SUSTAINABILITY_STATEMENT',
                'testMode': False
            },
            'authorizedBy': {
                'userId': authorized_by_user_id,
                'role': 'CHIEF_FINANCIAL_OFFICER',
                'timestamp': datetime.now().isoformat() + 'Z'
            },
            'submissionMetadata': {
                'lei': lei,
                'fiscalYear': '2024',
                'preparationDate': datetime.now().isoformat() + 'Z',
                'assuranceLevel': 'LIMITED'
            }
        }

        response = requests.post(
            f'{self.base_url}/submit/regulatory',
            json=payload,
            headers=self.headers
        )
        response.raise_for_status()
        return response.json()

    def poll_until_complete(self, report_job_id: str, timeout: int = 1800) -> Dict:
        """Poll report status until completion or timeout"""
        start_time = time.time()

        while time.time() - start_time < timeout:
            status = self.get_report_status(report_job_id)

            if status['status'] == 'COMPLETED':
                return status
            elif status['status'] == 'FAILED':
                raise Exception(f"Report generation failed: {status.get('error', 'Unknown error')}")

            time.sleep(30)  # Wait 30 seconds before next poll

        raise TimeoutError(f"Report generation timed out after {timeout} seconds")

# Usage Example
```

```python
def generate_csrd_climate_report():
    """Example usage of the compliance reporting client"""

    # Initialize client
    client = ComplianceReportingClient(access_token="your-access-token")

    # Step 1: Validate compliance first
    print("Validating compliance...")
    validation_result = client.validate_compliance(
        organization_id="org-123",
        standard="CSRD",
        template="ESRS_E1_CLIMATE_CHANGE",
        start_date=datetime(2024, 1, 1),
        end_date=datetime(2024, 12, 31)
    )

    compliance_score = validation_result['overallCompliance']['score']
    print(f"Compliance score: {compliance_score}%")

    if compliance_score < 80:
        print("Warning: Low compliance score. Consider addressing gaps before report generation.")
        for recommendation in validation_result['gapAnalysis']['recommendations']:
            print(f"- {recommendation}")

    # Step 2: Generate report
    print("Generating CSRD Climate Change report...")
    report_response = client.generate_report(
        organization_id="org-123",
        standard="CSRD",
        template="ESRS_E1_CLIMATE_CHANGE",
        start_date=datetime(2024, 1, 1),
        end_date=datetime(2024, 12, 31),
        output_formats=["PDF", "XLSX", "XML"]
    )

    report_job_id = report_response['reportJobId']
    print(f"Report generation started. Job ID: {report_job_id}")

    # Step 3: Poll for completion
    print("Waiting for report completion...")
    final_status = client.poll_until_complete(report_job_id)

    print(f"Report completed successfully!")
    print(f"Pages generated: {final_status['results']['totalPages']}")
    print(f"Data points processed: {final_status['results']['dataPointsProcessed']}")

    # Download URLs
```

```python
    for output in final_status['results']['outputs']:
        print(f"Download {output['format']}: {output['downloadUrl']}")

    # Step 4: Initiate approval workflow
    print("Initiating approval workflow...")
    approval_response = client.initiate_approval_workflow(
        report_job_id=report_job_id,
        organization_id="org-123"
    )

    approval_job_id = approval_response['approvalJobId']
    print(f"Approval workflow initiated. Job ID: {approval_job_id}")

    return {
        'report_job_id': report_job_id,
        'approval_job_id': approval_job_id,
        'download_urls': [output['downloadUrl'] for output in final_status['results']['outputs']]
    }

if __name__ == "__main__":
    try:
        result = generate_csrd_climate_report()
        print("Report generation process completed successfully!")
        print(f"Report Job ID: {result['report_job_id']}")
        print(f"Approval Job ID: {result['approval_job_id']}")
    except Exception as e:
        print(f"Error: {e}")
```

5.3 Node.js/TypeScript Integration

typescript

```typescript
import axios, { AxiosInstance, AxiosResponse } from 'axios';

interface ReportGenerationRequest {
  reportJobId: string;
  organizationId: string;
  reportConfig: {
    standard: string;
    template: string;
    reportingPeriod: {
      startDate: string;
      endDate: string;
      fiscalYear: string;
    };
    outputFormats: string[];
    language: string;
    currency: string;
  };
  dataFilters: {
    includeEstimatedData: boolean;
    qualityThreshold: number;
    includeThirdPartyData: boolean;
  };
}

interface ReportStatus {
  reportJobId: string;
  status: 'PROCESSING' | 'COMPLETED' | 'FAILED';
  progress?: {
    percentComplete: number;
    dataCollection: string;
    calculations: string;
    validation: string;
    formatting: string;
  };
  results?: {
    outputs: Array<{
      format: string;
      downloadUrl: string;
      fileSize: string;
    }>;
  };
}

export class ComplianceReportingClient {
  private client: AxiosInstance;
```

```
constructor(accessToken: string, baseURL: string = 'https://api.sweep.com/v1/compliance-reporting') {
  this.client = axios.create({
    baseURL,
    headers: {
      'Authorization': `Bearer ${accessToken}`,
      'Content-Type': 'application/json'
    }
  });
}

async generateReport(request: ReportGenerationRequest): Promise<any> {
  try {
    const response = await this.client.post('/reports/generate', request);
    return response.data;
  } catch (error) {
    throw this.handleError(error);
  }
}

async getReportStatus(reportJobId: string): Promise<ReportStatus> {
  try {
    const response = await this.client.get(`/reports/generate/status/${reportJobId}`);
    return response.data;
  } catch (error) {
    throw this.handleError(error);
  }
}

async validateCompliance(organizationId: string, standard: string, template: string): Promise<any> {
  try {
    const request = {
      validationJobId: `validation-job-${Date.now()}`,
      organizationId,
      validationConfig: {
        standard,
        template,
        strictMode: true,
        includeWarnings: true
      }
    };

    const response = await this.client.post('/validate/compliance', request);
    return response.data;
  } catch (error) {
    throw this.handleError(error);
  }
}
```

```
  async pollUntilComplete(reportJobId: string, timeoutMs: number = 1800000): Promise<ReportStatus> {
    const startTime = Date.now();
    const pollInterval = 30000; // 30 seconds

    while (Date.now() - startTime < timeoutMs) {
      const status = await this.getReportStatus(reportJobId);

      if (status.status === 'COMPLETED') {
        return status;
      } else if (status.status === 'FAILED') {
        throw new Error(`Report generation failed for job ${reportJobId}`);
      }

      await this.delay(pollInterval);
    }

    throw new Error(`Report generation timed out after ${timeoutMs}ms`);
  }

  private delay(ms: number): Promise<void> {
    return new Promise(resolve => setTimeout(resolve, ms));
  }

  private handleError(error: any): Error {
    if (error.response) {
      const errorData = error.response.data?.error;
      return new Error(`API Error: ${errorData?.message || error.message}`);
    }
    return new Error(`Network Error: ${error.message}`);
  }
}

// Usage Example
async function generateCSRDReport(): Promise<void> {
  const client = new ComplianceReportingClient('your-access-token');

  try {
    // Generate report
    const reportRequest: ReportGenerationRequest = {
      reportJobId: `report-job-${Date.now()}`,
      organizationId: 'org-123',
      reportConfig: {
        standard: 'CSRD',
        template: 'ESRS_E1_CLIMATE_CHANGE',
        reportingPeriod: {
          startDate: '2024-01-01T00:00:00Z',
```

```
        endDate: '2024-12-31T23:59:59Z',
        fiscalYear: '2024'
      },
      outputFormats: ['PDF', 'XLSX'],
      language: 'EN',
      currency: 'USD'
    },
    dataFilters: {
      includeEstimatedData: true,
      qualityThreshold: 85.0,
      includeThirdPartyData: true
    }
  };

  const reportResponse = await client.generateReport(reportRequest);
  console.log('Report generation started:', reportResponse.reportJobId);

  // Poll for completion
  const finalStatus = await client.pollUntilComplete(reportResponse.reportJobId);
  console.log('Report completed successfully!');

  // Display download URLs
  finalStatus.results?.outputs.forEach(output => {
    console.log(`Download ${output.format}: ${output.downloadUrl}`);
  });

} catch (error) {
  console.error('Error generating report:', error);
}
}
```

6. Webhooks & Event Notifications

6.1 Webhook Configuration

POST /webhooks/register

- **Purpose:** Register webhook endpoints for event notifications

Request Body:

```
json
```

```
{
  "webhookId": "webhook-123",
  "organizationId": "org-123",
  "callbackUrl": "https://client.sweep.com/webhooks/compliance-events",
  "events": [
    "REPORT_GENERATION_COMPLETED",
    "REPORT_GENERATION_FAILED",
    "APPROVAL_WORKFLOW_STATUS_CHANGED",
    "REGULATORY_SUBMISSION_COMPLETED"
  ],
  "headers": {
    "Authorization": "Bearer client-webhook-token",
    "X-Client-Secret": "webhook-secret-key"
  },
  "retryPolicy": {
    "maxRetries": 3,
    "retryDelay": "30s"
  }
}
```

6.2 Webhook Event Payloads

Report Generation Completed:

json

```
{
  "eventId": "event-001",
  "eventType": "REPORT_GENERATION_COMPLETED",
  "timestamp": "2025-09-25T16:18:45Z",
  "organizationId": "org-123",
  "data": {
    "reportJobId": "report-job-abc123",
    "standard": "CSRD",
    "template": "ESRS_E1_CLIMATE_CHANGE",
    "status": "COMPLETED",
    "downloadUrls": [
      "/reports/download/report-job-abc123.pdf",
      "/reports/download/report-job-abc123.xlsx"
    ],
    "complianceScore": 97.6
  }
}
```

7. Rate Limits & Quotas

7.1 API Rate Limits

Endpoint Category	Rate Limit	Quota Period
Report Generation	10 requests/hour	Per organization
Status Checking	100 requests/hour	Per organization
Template Retrieval	1000 requests/hour	Per organization
Validation	50 requests/hour	Per organization
Audit Trail	200 requests/hour	Per organization

7.2 Resource Quotas

- **Monthly Report Generation:** 100 reports per organization
- **Storage Quota:** 50GB per organization
- **File Retention:** 7 years for compliance reports
- **Concurrent Processing:** 5 simultaneous report generations per organization

8. Security & Compliance

8.1 Data Security

- **Encryption in Transit:** TLS 1.3 for all API communications
- **Encryption at Rest:** AES-256 for stored reports and data
- **Access Controls:** Role-based permissions with principle of least privilege
- **Audit Logging:** Complete audit trail for all operations

8.2 Regulatory Compliance

- **GDPR Compliance:** Data processing lawfulness and subject rights
- **SOX Compliance:** Financial reporting controls and documentation
- **ISO 27001:** Information security management
- **SOC 2 Type II:** Security, availability, and confidentiality controls

8.3 Digital Signatures & Certificates

- **Report Signing:** Support for qualified digital signatures
- **Certificate Management:** Integration with PKI infrastructure
- **Regulatory Recognition:** Compliance with eIDAS regulation

9. Monitoring & Observability

9.1 Health Endpoints

- **GET /health:** Service health check
- **GET /health/deep:** Comprehensive system health
- **GET /metrics:** Prometheus-compatible metrics
- **GET /info:** Service version and build information

9.2 Key Metrics

- **Report Generation Success Rate:** Target >99.5%
- **Average Processing Time:** Target <15 minutes
- **API Availability:** Target >99.9%
- **Compliance Validation Accuracy:** Target >99.8%

10. Versioning & Migration

10.1 API Versioning Strategy

- **Version Format:** Semantic versioning (v1.0.0)
- **Backward Compatibility:** Minimum 12 months support for deprecated versions
- **Migration Path:** Gradual deprecation with advance notice

10.2 Template Versioning

- **Automatic Updates:** Templates updated with regulatory changes
- **Version Control:** Full history of template modifications
- **Migration Support:** Automated migration of existing reports to new template versions

11. Support & Troubleshooting

11.1 Common Issues

Issue	Cause	Resolution
Report Generation Timeout	Large data volume or complex calculations	Optimize data query scope or split into smaller reports
Validation Failures	Missing required data points	Review data completeness using validation endpoint
Template Errors	Outdated template version	Update to latest template version
Submission Failures	Regulatory portal connectivity	Retry after checking portal status

11.2 Support Channels

- **Technical Documentation:** docs.sweep.com/compliance-reporting
- **API Support:** compliance-api-support@sweep.com
- **Regulatory Updates:** regulatory-updates@sweep.com
- **Emergency Support:** 24/7 support for production issues

12. Appendices

Appendix A: Supported Regulatory Standards

Standard	Version	Effective Date	Mandatory For
CSRD	2024.1	2025-01-01	Large EU companies
SFDR	2024.2	2021-03-10	EU financial market participants
ISSB S1/S2	2023.1	2024-01-01	Global (voluntary adoption)
GRI Universal Standards	2021	2023-01-01	Global (voluntary)
CDP Climate Change	2024	2024-01-01	Responding organizations
GHG Protocol	2004	Ongoing	Global standard

Appendix B: Data Point Mappings

```json
{
  "scope_1_emissions": {
    "description": "Direct GHG emissions from owned or controlled sources",
    "unit": "tonnes CO2e",
    "calculationMethod": "SUM",
    "regulatoryReferences": ["CSRD:ESRS-E1-6", "GHG:Scope1"]
  },
  "renewable_energy_percentage": {
    "description": "Percentage of energy consumption from renewable sources",
    "unit": "percentage",
    "calculationMethod": "WEIGHTED_AVERAGE",
    "regulatoryReferences": ["CSRD:ESRS-E1-5", "GRI:302-1"]
  }
}
```

Appendix C: Error Code Reference

Complete list of all error codes, their meanings, and recommended actions for resolution.

Document Version Control:

- v1.0.0 - Initial version (2025-09-25)
- Authors: Shireen Hussain J, Product Architecture Team
- Reviewers: ESG Platform Team, Compliance Team
- Next Review Date: 2025-12-25