System Configuration API Functions

Technical Manual - API Reference

SONARWORKS WORKFLOW SYSTEM

Version 1.1

Table of Contents

1. Introduction

This manual documents the System Configuration API functions for the Sonar Workflow System. These APIs enable programmatic management of system settings, email configuration, audit logs, and other administrative functions.

API Scope:

- System settings management
- Email server configuration
- Audit log queries
- Data import/export
- Report generation
- System health monitoring

2. Settings API Endpoints

Method	Endpoint	Description
GET	/api/settings	Get all settings
GET	/api/settings/{key}	Get setting by key
PUT	/api/settings/{key}	Update setting
POST	/api/settings	Create/update multiple settings

2.1 Get All Settings

```http
GET /api/settings
```

Query Parameters:

Parameter	Type	Description
module	string	Filter by module (e.g., WORKFLOW, EMAIL, SECURITY)

Response:

```json
{
  "success": true,
  "data": [
    {
      "id": "setting-uuid-1",
      "key": "system.name",
      "value": "Sonar Workflow System",
      "type": "STRING",
      "module": "SYSTEM",
```

```
      "description": "System display name"
    },
    {
      "id": "setting-uuid-2",
      "key": "workflow.require_approvers",
      "value": "true",
      "type": "BOOLEAN",
      "module": "WORKFLOW",
      "description": "Require at least one approver for workflows"
    },
    {
      "id": "setting-uuid-3",
      "key": "security.session_timeout",
      "value": "30",
      "type": "NUMBER",
      "module": "SECURITY",
      "description": "Session timeout in minutes"
    }
  ]
}
```

2.2 Get Setting by Key

```
[HTTP]
GET /api/settings/{key}
```

Example:

```
[HTTP]
GET /api/settings/system.name
```

Response:

```
[JSON]
{
  "success": true,
  "data": {
    "key": "system.name",
    "value": "Sonar Workflow System",
    "type": "STRING"
  }
}
```

2.3 Update Setting

```
[HTTP]
PUT /api/settings/{key}
```

Request Body:

```
[JSON]
{
  "value": "My Custom Workflow System"
}
```

2.4 Batch Update Settings

```
[HTTP]
POST /api/settings
```

Request Body:

```
[JSON]
{
  "settings": [
    {"key": "system.name", "value": "Updated System Name"},
    {"key": "security.session_timeout", "value": "60"},
    {"key": "workflow.require_comment", "value": "true"}
  ]
}
```

2.5 Common Settings Reference

Key	Type	Description
system.name	STRING	System display name
system.timezone	STRING	Default timezone
workflow.require_approvers	BOOLEAN	Require approvers
workflow.require_comment	BOOLEAN	Require approval comments
security.session_timeout	NUMBER	Session timeout (minutes)
security.max_failed_logins	NUMBER	Max failed login attempts
security.lockout_duration	NUMBER	Account lockout duration (minutes)

3. Email Configuration API

Method	Endpoint	Description
GET	/api/email-settings	Get email configuration
PUT	/api/email-settings	Update email configuration
POST	/api/email-settings/test	Test email configuration

3.1 Get Email Configuration

```
[HTTP]
GET /api/email-settings
```

Response:

```
[JSON]
{
```

```
  "success": true,
  "data": {
    "id": "email-settings-uuid",
    "host": "smtp.company.com",
    "port": 587,
    "protocol": "SMTP",
    "username": "workflow@company.com",
    "password": "********",
    "fromAddress": "noreply@company.com",
    "fromName": "Workflow System",
    "enableTls": true,
    "enableSsl": false,
    "isConfigured": true,
    "lastTestResult": {
      "success": true,
      "testedAt": "2024-01-15T10:00:00Z"
    }
  }
}
```

3.2 Update Email Configuration

[HTTP]

```
PUT /api/email-settings
```

Request Body:

[JSON]

```
{
  "host": "smtp.company.com",
  "port": 587,
  "protocol": "SMTP",
  "username": "workflow@company.com",
  "password": "SecurePassword123",
  "fromAddress": "noreply@company.com",
  "fromName": "Workflow System",
  "enableTls": true,
  "enableSsl": false
}
```

3.3 Test Email Configuration

[HTTP]

```
POST /api/email-settings/test
```

Request Body:

[JSON]

```
{
```

```
  "recipientEmail": "test@company.com"
}
```

Response:

```
[JSON]
{
  "success": true,
  "message": "Test email sent successfully",
  "data": {
    "sentTo": "test@company.com",
    "sentAt": "2024-01-15T10:30:00Z"
  }
}
```

4. Audit Log API

Method	Endpoint	Description
GET	/api/audit	Query audit logs
GET	/api/audit/{id}	Get audit entry by ID
GET	/api/audit/export	Export audit logs

4.1 Query Audit Logs

```
[HTTP]
GET /api/audit
```

Query Parameters:

Parameter	Type	Description
page	integer	Page number
size	integer	Items per page
fromDate	datetime	Start date (ISO 8601)
toDate	datetime	End date (ISO 8601)
userId	string	Filter by user ID
username	string	Filter by username
action	string	Filter by action type
entityType	string	Filter by entity type
entityId	string	Filter by specific entity
module	string	Filter by module
ipAddress	string	Filter by IP address

Response:

```
[JSON]
{
```

```
  "success": true,
  "data": {
    "content": [
      {
        "id": "audit-uuid-1",
        "timestamp": "2024-01-15T14:30:25Z",
        "user": {
          "id": "user-uuid",
          "username": "john.doe"
        },
        "action": "UPDATE",
        "entityType": "User",
        "entityId": "target-user-uuid",
        "summary": "Updated user profile",
        "module": "USER_MANAGEMENT",
        "changes": {
          "email": {
            "oldValue": "old@email.com",
            "newValue": "new@email.com"
          }
        },
        "ipAddress": "192.168.1.100",
        "userAgent": "Mozilla/5.0...",
        "sessionId": "session-uuid"
      }
    ],
    "totalElements": 1000,
    "totalPages": 100,
    "number": 0
  }
}
```


4.2 Export Audit Logs

```
[HTTP]
GET /api/audit/export
```

Query Parameters:

Parameter	Type	Description
format	string	Export format (CSV, EXCEL, PDF)
fromDate	datetime	Start date
toDate	datetime	End date
...other filters		Same as query endpoint

Response: Binary file download

5. Import/Export API

Method	Endpoint	Description
POST	/api/import/users	Import users from file
POST	/api/import/workflows	Import workflow definitions
GET	/api/export/users	Export users
GET	/api/export/workflows	Export workflows
GET	/api/export/workflow-instances	Export instances

5.1 Import Users

```
[HTTP]
POST /api/import/users
Content-Type: multipart/form-data
```

Request: Multipart form with CSV file

CSV Format:

```
[CSV]
username,email,firstName,lastName,userType,roles
john.doe,john@company.com,John,Doe,STAFF,"Staff,Initiator"
jane.smith,jane@company.com,Jane,Smith,MANAGER,"Manager,Approver"
```

Response:

```
[JSON]
{
  "success": true,
  "message": "Import completed",
  "data": {
    "totalRows": 50,
    "imported": 48,
    "failed": 2,
    "errors": [
      {"row": 23, "error": "Email already exists"},
      {"row": 45, "error": "Invalid user type"}
    ]
  }
}
```

5.2 Export Workflow Instances

```
[HTTP]
GET /api/export/workflow-instances
```

Query Parameters:

Parameter	Type	Description
format	string	EXCEL, CSV, PDF
workflowCode	string	Filter by workflow
status	string	Filter by status
fromDate	date	Submission date from
toDate	date	Submission date to

6. Report API

Method	Endpoint	Description
GET	/api/reports	List available reports
GET	/api/reports/{code}	Get report metadata
POST	/api/reports/{code}/generate	Generate report
GET	/api/reports/{code}/download	Download report

6.1 List Reports

[HTTP]
```
GET /api/reports
```

Response:

[JSON]
```
{
  "success": true,
  "data": [
    {
      "code": "WORKFLOW_SUMMARY",
      "name": "Workflow Summary Report",
      "description": "Overview of all workflow submissions",
      "category": "WORKFLOW",
      "parameters": [
        {"name": "fromDate", "type": "DATE", "required": true},
        {"name": "toDate", "type": "DATE", "required": true},
        {"name": "status", "type": "SELECT", "required": false}
      ]
    },
    {
      "code": "USER_ACTIVITY",
      "name": "User Activity Report",
      "description": "User login and action summary",
      "category": "USER",
      "parameters": [...]
    }
  ]
}
```

6.2 Generate Report

```
[HTTP]
POST /api/reports/{code}/generate
```

Request Body:

```
[JSON]
{
  "parameters": {
    "fromDate": "2024-01-01",
    "toDate": "2024-01-31",
    "status": "APPROVED"
  },
  "format": "EXCEL"
}
```

Response:

```
[JSON]
{
  "success": true,
  "data": {
    "reportId": "report-run-uuid",
    "status": "COMPLETED",
    "generatedAt": "2024-01-15T15:00:00Z",
    "downloadUrl": "/api/reports/WORKFLOW_SUMMARY/download?id=report-
run-uuid",
    "summary": {
      "totalRecords": 245,
      "generationTime": "2.3s"
    }
  }
}
```

7. Category API

Method	Endpoint	Description
GET	/api/categories	List all categories
GET	/api/categories/{id}	Get category by ID
POST	/api/categories	Create new category
PUT	/api/categories/{id}	Update category
DELETE	/api/categories/{id}	Delete category

7.1 Category Data Structure

```
[JSON]
{
```

```
  "id": "category-uuid",
  "name": "Financial Services",
  "description": "Banks, insurance, and investment companies",
  "isActive": true,
  "corporateCount": 5
}
```

8. Health Check API

Method	Endpoint	Description
GET	/api/health	Basic health check
GET	/api/health/detailed	Detailed health check
GET	/api/health/db	Database connection check
GET	/api/health/email	Email service check

8.1 Basic Health Check

```
[HTTP]
GET /api/health
```

Response:

```
[JSON]
{
  "status": "UP",
  "timestamp": "2024-01-15T15:30:00Z"
}
```

8.2 Detailed Health Check

```
[HTTP]
GET /api/health/detailed
```

Response:

```
[JSON]
{
  "status": "UP",
  "timestamp": "2024-01-15T15:30:00Z",
  "components": {
    "database": {
      "status": "UP",
      "details": {
        "type": "PostgreSQL",
        "version": "14.5"
      }
    },
    "email": {
```

```
      "status": "UP",
      "details": {
        "host": "smtp.company.com",
        "lastTest": "2024-01-15T10:00:00Z"
      }
    },
    "cache": {
      "status": "UP",
      "details": {
        "entries": 1245,
        "hitRate": "0.85"
      }
    }
  },
  "system": {
    "uptime": "5d 3h 25m",
    "version": "1.0.0",
    "javaVersion": "17.0.5",
    "memory": {
      "used": "512MB",
      "max": "2048MB"
    }
  }
}
```

9. Error Handling

Error Code	HTTP Status	Description
SETTING_NOT_FOUND	404	Setting key does not exist
INVALID_SETTING_VALUE	422	Value type mismatch
EMAIL_CONFIG_ERROR	400	Invalid email configuration
EMAIL_TEST_FAILED	500	Email test could not be sent
IMPORT_FAILED	400	Import file validation failed
EXPORT_FAILED	500	Export generation failed
REPORT_NOT_FOUND	404	Report does not exist

10. Code Examples

10.1 JavaScript Settings Management

[JavaScript]

```
class ConfigService {
  constructor(apiUrl, token) {
    this.apiUrl = apiUrl;
    this.headers = {
```

```
      'Authorization': `Bearer ${token}`,
      'Content-Type': 'application/json'
    };
  }

  async getSettings(module = null) {
    const params = module ? `?module=${module}` : '';
    const response = await fetch(`${this.apiUrl}/api/settings${params}`,
{
      headers: this.headers
    });
    return response.json();
  }

  async updateSetting(key, value) {
    const response = await fetch(`${this.apiUrl}/api/settings/${key}`, {
      method: 'PUT',
      headers: this.headers,
      body: JSON.stringify({ value })
    });
    return response.json();
  }

  async testEmailConfig(recipientEmail) {
    const response = await fetch(`${this.apiUrl}/api/email-
settings/test`, {
      method: 'POST',
      headers: this.headers,
      body: JSON.stringify({ recipientEmail })
    });
    return response.json();
  }

  async healthCheck() {
    const response = await fetch(`${this.apiUrl}/api/health/detailed`, {
      headers: this.headers
    });
    return response.json();
  }
}

// Usage
const config = new ConfigService('https://api.server.com', token);

// Get all settings
const settings = await config.getSettings();

// Update session timeout
```

```
await config.updateSetting('security.session_timeout', '60');

// Test email
const emailTest = await config.testEmailConfig('admin@company.com');
```

10.2 Python Audit Log Query

[Python]

```
import requests
from datetime import datetime, timedelta

class AuditClient:
    def __init__(self, base_url, token):
        self.base_url = base_url
        self.headers = {"Authorization": f"Bearer {token}"}

    def query_logs(self, **params):
        response = requests.get(
            f"{self.base_url}/api/audit",
            headers=self.headers,
            params=params
        )
        return response.json()

    def export_logs(self, format="CSV", **params):
        params["format"] = format
        response = requests.get(
            f"{self.base_url}/api/audit/export",
            headers=self.headers,
            params=params,
            stream=True
        )
        return response.content

# Usage
client = AuditClient("https://api.server.com", token)

# Query recent logins
today = datetime.now().strftime("%Y-%m-%d")
week_ago = (datetime.now() - timedelta(days=7)).strftime("%Y-%m-%d")

logins = client.query_logs(
    action="LOGIN",
    fromDate=week_ago,
    toDate=today,
    size=100
)

for entry in logins["data"]["content"]:
```

```
    print(f"{entry['timestamp']}: {entry['user']['username']} from
{entry['ipAddress']}")

# Export to CSV
csv_data = client.export_logs(
    format="CSV",
    fromDate=week_ago,
    toDate=today
)
with open("audit_export.csv", "wb") as f:
    f.write(csv_data)
```