User & Organization API Functions

Technical Manual - API Reference

SONARWORKS WORKFLOW SYSTEM

Version 1.0

Table of Contents

1. Introduction

This manual documents the User and Organization API functions for the Sonarworks Workflow System. These APIs enable programmatic management of users, roles, and organizational structures.

API Scope:

- User management (CRUD operations)
- Role and privilege management
- Corporate entity management
- SBU management
- Branch management
- Department management

NOTE: Most endpoints require administrative privileges.

2. User API Endpoints

Method	Endpoint	Description
GET	/api/users	List all users
GET	/api/users/{id}	Get user by ID
POST	/api/users	Create new user
PUT	/api/users/{id}	Update user
DELETE	/api/users/{id}	Delete user
POST	/api/users/{id}/lock	Lock user account
POST	/api/users/{id}/unlock	Unlock user account
POST	/api/users/{id}/reset-password	Reset user password

3. User CRUD Operations

3.1 List Users

```
[HTTP]
GET /api/users
```

Query Parameters:

Parameter	Type	Description
page	integer	Page number (0-indexed)
size	integer	Items per page
search	string	Search in username, name, email
userType	string	Filter by type (SYSTEM, STAFF, MANAGER, EXTERNAL)
status	string	Filter by status (ACTIVE, INACTIVE, LOCKED)

roleId	string	Filter by role assignment

Response:

```
[JSON]
{
  "success": true,
  "data": {
    "content": [
      {
        "id": "user-uuid-1",
        "username": "john.doe",
        "email": "john.doe@company.com",
        "firstName": "John",
        "lastName": "Doe",
        "phone": "+1234567890",
        "staffId": "EMP001",
        "userType": "STAFF",
        "isActive": true,
        "isLocked": false,
        "roles": [
          {"id": "role-uuid-1", "name": "Staff"},
          {"id": "role-uuid-2", "name": "Initiator"}
        ],
        "lastLogin": "2024-01-15T09:30:00Z",
        "createdAt": "2023-06-01T10:00:00Z"
      }
    ],
    "totalElements": 150,
    "totalPages": 15,
    "number": 0
  }
}
```

3.2 Get User by ID

```
[HTTP]
GET /api/users/{userId}
```

Response:

```
[JSON]
{
  "success": true,
  "data": {
    "id": "user-uuid-1",
    "username": "john.doe",
    "email": "john.doe@company.com",
    "firstName": "John",
    "lastName": "Doe",
    "phone": "+1234567890",
    "staffId": "EMP001",
    "userType": "STAFF",
    "isActive": true,
    "isLocked": false,
    "lockReason": null,
    "mustChangePassword": false,
    "failedLoginAttempts": 0,
    "lastLogin": "2024-01-15T09:30:00Z",
    "roles": [
      {"id": "role-uuid-1", "name": "Staff"},
      {"id": "role-uuid-2", "name": "Initiator"}
    ],
    "corporates": [
      {"id": "corporate-uuid", "name": "ABC Holdings"}
    ],
    "sbus": [
      {"id": "sbu-uuid", "name": "Finance"}
    ],
    "branches": [
      {"id": "branch-uuid", "name": "Head Office"}
    ],
    "departments": [
      {"id": "dept-uuid", "name": "IT"}
    ],
    "createdAt": "2023-06-01T10:00:00Z",
    "updatedAt": "2024-01-10T14:30:00Z"
  }
}
```

3.3 Create User

[HTTP]

```
POST /api/users
```

Request Body:

[JSON]

```
{
  "username": "jane.smith",
  "email": "jane.smith@company.com",
  "firstName": "Jane",
  "lastName": "Smith",
  "phone": "+1234567891",
  "staffId": "EMP002",
  "userType": "STAFF",
  "password": "SecureP@ss123",
  "mustChangePassword": true,
  "roleIds": ["role-uuid-1", "role-uuid-2"],
  "corporateIds": ["corporate-uuid"],
  "sbuIds": ["sbu-uuid"],
  "branchIds": ["branch-uuid"],
  "departmentIds": ["dept-uuid"]
}
```

Response:

[JSON]
```
{
  "success": true,
  "message": "User created successfully",
  "data": {
    "id": "new-user-uuid",
    "username": "jane.smith"
  }
}
```

3.4 Update User

[HTTP]
```
PUT /api/users/{userId}
```

Request Body (partial update supported):

[JSON]
```
{
  "firstName": "Jane",
  "lastName": "Smith-Johnson",
  "phone": "+1234567892",
  "roleIds": ["role-uuid-1", "role-uuid-3"]
}
```

3.5 Delete User

[HTTP]
```
DELETE /api/users/{userId}
```

WARNING: Deleting a user may fail if they have associated workflow instances. Consider deactivating instead.

3.6 Lock User

[HTTP]
```
POST /api/users/{userId}/lock
```

Request Body:

[JSON]
```
{
  "reason": "Security investigation pending"
}
```

3.7 Unlock User

[HTTP]
```
POST /api/users/{userId}/unlock
```

3.8 Reset Password

[HTTP]
```
POST /api/users/{userId}/reset-password
```

Request Body:

[JSON]
```
{
  "newPassword": "TempP@ss123",
  "mustChangePassword": true
}
```

4. Role API Endpoints

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

4.1 List Roles

[HTTP]
```
GET /api/roles
```

Response:

[JSON]
```
{
  "success": true,
  "data": [
    {
      "id": "role-uuid-1",
      "name": "Administrator",
      "description": "Full system access",
      "isSystemRole": true,
      "privileges": [
        {"id": "priv-uuid-1", "name": "MANAGE_USERS"},
        {"id": "priv-uuid-2", "name": "MANAGE_WORKFLOWS"}
      ],
      "userCount": 3
    },
    {
      "id": "role-uuid-2",
      "name": "Initiator",
      "description": "Submit and track workflows",
      "isSystemRole": true,
      "privileges": [
        {"id": "priv-uuid-3", "name": "SUBMIT_WORKFLOW"},
        {"id": "priv-uuid-4", "name": "VIEW_OWN_SUBMISSIONS"}
      ],
      "userCount": 45
    }
  ]
}
```

4.2 Create Role

[HTTP]
```
POST /api/roles
```

Request Body:

[JSON]
```
{
  "name": "Finance Reviewer",
  "description": "Review and approve financial workflows",
  "privilegeIds": ["priv-uuid-1", "priv-uuid-2", "priv-uuid-3"]
}
```

4.3 List Privileges

[HTTP]
```
GET /api/privileges
```

Response:

[JSON]
```
{
  "success": true,
  "data": [
    {"id": "priv-uuid-1", "name": "MANAGE_USERS", "description":
"Create/edit/delete users"},
    {"id": "priv-uuid-2", "name": "MANAGE_WORKFLOWS", "description":
"Create/edit workflows"},
    {"id": "priv-uuid-3", "name": "SUBMIT_WORKFLOW", "description":
"Submit workflow instances"},
    {"id": "priv-uuid-4", "name": "APPROVE_WORKFLOW", "description":
```

```
"Approve workflow instances"},
    {"id": "priv-uuid-5", "name": "VIEW_REPORTS", "description": "Access
reports module"}
  ]
}
```

5. Corporate API Endpoints

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

5.1 Corporate Data Structure

[JSON]

```json
{
  "id": "corporate-uuid",
  "code": "ABC",
  "name": "ABC Holdings Ltd",
  "category": {
    "id": "category-uuid",
    "name": "Financial Services"
  },
  "type": "HOLDING",
  "address": "123 Business Street",
  "email": "info@abc.com",
  "phone": "+1234567890",
  "website": "https://www.abc.com",
  "isActive": true,
  "createdAt": "2023-01-01T00:00:00Z"
}
```

5.2 Create Corporate

[JSON]

```json
{
  "code": "XYZ",
  "name": "XYZ Subsidiary Inc",
  "categoryId": "category-uuid",
  "type": "SUBSIDIARY",
  "address": "456 Corporate Drive",
  "email": "contact@xyz.com",
  "phone": "+1234567891"
}
```

6. SBU API Endpoints

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

6.1 SBU Data Structure

[JSON]

```json
{
  "id": "sbu-uuid",
  "code": "FIN-001",
  "name": "Finance Division",
  "corporate": {
    "id": "corporate-uuid",
    "code": "ABC",
    "name": "ABC Holdings Ltd"
  },
  "parentSbu": null,
  "address": "123 Finance Street",
  "email": "finance@abc.com",
  "phone": "+1234567892",
  "isActive": true,
  "childSbus": [
    {
      "id": "child-sbu-uuid",
      "code": "FIN-002",
      "name": "Retail Banking"
    }
  ]
}
```

6.2 Create SBU

[JSON]

```json
{
  "code": "OPS-001",
  "name": "Operations Division",
  "corporateId": "corporate-uuid",
  "parentSbuId": null,
  "address": "789 Operations Way",
  "email": "ops@abc.com"
}
```

7. Branch API Endpoints

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

7.1 Branch Data Structure

[JSON]

```json
{
  "id": "branch-uuid",
  "code": "HQ-001",
  "name": "Head Office",
  "sbu": {
    "id": "sbu-uuid",
    "code": "FIN-001",
    "name": "Finance Division"
  },
  "address": "100 Main Street",
  "city": "New York",
  "phone": "+1234567893",
  "email": "headoffice@abc.com",
  "isActive": true
}
```

8. Department API Endpoints

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

8.1 Department Data Structure

[JSON]

```json
{
  "id": "dept-uuid",
  "code": "IT-001",
  "name": "Information Technology",
  "branch": {
    "id": "branch-uuid",
    "code": "HQ-001",
    "name": "Head Office"
  },
  "email": "it@abc.com",
  "phone": "+1234567894",
  "isActive": true
}
```

9. Error Handling

Error Code	HTTP Status	Description
USER_NOT_FOUND	404	User does not exist
USERNAME_EXISTS	409	Username already taken
EMAIL_EXISTS	409	Email already registered
ROLE_NOT_FOUND	404	Role does not exist
CORPORATE_NOT_FOUND	404	Corporate does not exist
SBU_NOT_FOUND	404	SBU does not exist
BRANCH_NOT_FOUND	404	Branch does not exist
DEPARTMENT_NOT_FOUND	404	Department does not exist
CANNOT_DELETE	400	Entity has dependencies
VALIDATION_ERROR	422	Field validation failed

10. Code Examples

10.1 JavaScript User Management

[JavaScript]

```
class UserService {
  constructor(apiUrl, token) {
    this.apiUrl = apiUrl;
    this.headers = {
      'Authorization': `Bearer ${token}`,
      'Content-Type': 'application/json'
    };
  }

  async listUsers(params = {}) {
    const query = new URLSearchParams(params).toString();
    const response = await fetch(`${this.apiUrl}/api/users?${query}`, {
      headers: this.headers
    });
    return response.json();
  }

  async createUser(userData) {
    const response = await fetch(`${this.apiUrl}/api/users`, {
      method: 'POST',
      headers: this.headers,
      body: JSON.stringify(userData)
    });
    return response.json();
  }

  async updateUser(userId, userData) {
    const response = await fetch(`${this.apiUrl}/api/users/${userId}`, {
      method: 'PUT',
      headers: this.headers,
      body: JSON.stringify(userData)
    });
    return response.json();
  }

  async lockUser(userId, reason) {
    const response = await
fetch(`${this.apiUrl}/api/users/${userId}/lock`, {
      method: 'POST',
      headers: this.headers,
      body: JSON.stringify({ reason })
    });
    return response.json();
  }
}

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

// Create user
const newUser = await userService.createUser({
  username: 'new.user',
  email: 'new.user@company.com',
  firstName: 'New',
  lastName: 'User',
  userType: 'STAFF',
  password: 'TempP@ss123',
  mustChangePassword: true,
  roleIds: ['initiator-role-id']
});
```

10.2 Python Organization Management

[Python]

```
import requests

class OrgClient:
    def __init__(self, base_url, token):
        self.base_url = base_url
        self.headers = {
            "Authorization": f"Bearer {token}",
            "Content-Type": "application/json"
        }

    def list_corporates(self):
        return requests.get(
            f"{self.base_url}/api/corporates",
            headers=self.headers
        ).json()

    def create_sbu(self, sbu_data):
        return requests.post(
            f"{self.base_url}/api/sbus",
            headers=self.headers,
            json=sbu_data
        ).json()

    def create_branch(self, branch_data):
        return requests.post(
            f"{self.base_url}/api/branches",
            headers=self.headers,
            json=branch_data
        ).json()

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

# Create organizational hierarchy
sbu = client.create_sbu({
    "code": "NEW-SBU",
    "name": "New Business Unit",
    "corporateId": "corporate-uuid"
})

branch = client.create_branch({
    "code": "NEW-BR",
    "name": "New Branch",
    "sbuId": sbu["data"]["id"]
})
```