Approval API Functions

Technical Manual - API Reference

SONARWORKS WORKFLOW SYSTEM

Version 1.1

Table of Contents

1. Introduction

This manual documents the Approval API functions for the Sonar Workflow System. These APIs enable programmatic management of workflow approval processes including approving, rejecting, and escalating workflow instances.

API Capabilities:

- Retrieve pending approvals
- Process approval decisions (approve/reject/escalate)
- Add comments to approval actions
- Handle email-based approvals
- Track approval history

2. Approval Endpoints Overview

Method	Endpoint	Description
GET	/api/approvals	List pending approvals
GET	/api/approvals/{id}	Get approval details
POST	/api/approvals/{id}/approve	Approve workflow
POST	/api/approvals/{id}/reject	Reject workflow
POST	/api/approvals/{id}/escalate	Escalate workflow
POST	/api/email-approval/verify	Verify email approval token
POST	/api/email-approval/process	Process email approval action

3. List Pending Approvals

3.1 Endpoint

```
[HTTP]
GET /api/approvals
```

3.2 Query Parameters

Parameter	Type	Description
page	integer	Page number (0-indexed)
size	integer	Items per page (default: 20)
sort	string	Sort field and direction
workflowCode	string	Filter by workflow type
sbuId	string	Filter by SBU
status	string	PENDING, ESCALATED

3.3 Response

[JSON]


```
{
  "success": true,
  "data": {
    "content": [
      {
        "id": "instance-uuid-1",
        "referenceNumber": "WF-2024-00001",
        "workflow": {
          "id": "workflow-uuid",
          "code": "PURCHASE_REQUEST",
          "name": "Purchase Request"
        },
        "initiator": {
          "id": "user-uuid",
          "username": "john.doe",
          "fullName": "John Doe",
          "department": "IT"
        },
        "status": "PENDING",
        "currentLevel": 1,
        "amount": 2500.00,
        "sbu": {
          "id": "sbu-uuid",
          "name": "Finance"
        },
        "submittedAt": "2024-01-15T10:30:00Z",
        "daysInQueue": 2,
        "summary": "Office supplies request for Q1"
      }
    ],
    "totalElements": 15,
    "totalPages": 2,
    "number": 0
  }
}
```


4. Get Approval Details

4.1 Endpoint

[HTTP]

```
GET /api/approvals/{instanceId}
```

4.2 Response

[JSON]


```
{
  "success": true,
  "data": {
    "id": "instance-uuid",
    "referenceNumber": "WF-2024-00001",
    "workflow": {
      "id": "workflow-uuid",
      "code": "PURCHASE_REQUEST",
      "name": "Purchase Request",
      "requireComment": true
    },
    "initiator": {
      "id": "user-uuid",
      "username": "john.doe",
      "fullName": "John Doe",
      "email": "john.doe@company.com",
      "department": "IT"
    },
    "status": "PENDING",
    "currentLevel": 1,
    "amount": 2500.00,
    "sbu": {
      "id": "sbu-uuid",
      "name": "Finance"
    },
    "fieldValues": [
      {
        "field": {
          "id": "field-uuid-1",
          "name": "itemDescription",
          "label": "Item Description",
          "fieldType": "TEXT"
        },
        "value": "Office supplies for Q1 2024",
        "displayValue": "Office supplies for Q1 2024"
      },
      {
        "field": {
          "id": "field-uuid-2",
          "name": "amount",
          "label": "Total Amount",
          "fieldType": "CURRENCY"
        },
        "value": "2500.00",
        "displayValue": "$2,500.00"
```

```
      },
      {
        "field": {
          "id": "field-uuid-3",
          "name": "vendor",
          "label": "Vendor",
          "fieldType": "TEXT"
        },
        "value": "Office Depot",
        "displayValue": "Office Depot"
      }
    ],
    "attachments": [
      {
        "id": "attachment-uuid",
        "fileName": "quotation.pdf",
        "contentType": "application/pdf",
        "fileSize": 102400,
        "downloadUrl": "/api/attachments/attachment-uuid/download"
      }
    ],
    "approvalHistory": [
      {
        "id": "history-uuid-1",
        "action": "SUBMITTED",
        "level": 0,
        "actor": {
          "id": "user-uuid",
          "fullName": "John Doe"
        },
        "actionDate": "2024-01-15T10:30:00Z",
        "comments": "Initial submission",
        "source": "SYSTEM"
      }
    ],
    "submittedAt": "2024-01-15T10:30:00Z",
    "canApprove": true,
    "canReject": true,
    "canEscalate": true
  }
}
```

5. Approve Workflow

5.1 Endpoint

[HTTP]

```
POST /api/approvals/{instanceId}/approve
```

5.2 Request Body

[JSON]

```
{
  "comments": "Reviewed and approved. Budget confirmed."
}
```

Request Parameters:

Parameter	Type	Required	Description
comments	string	Conditional	Required if workflow requireComment=true

5.3 Response

[JSON]

```
{
  "success": true,
  "message": "Workflow approved successfully",
  "data": {
    "id": "instance-uuid",
    "referenceNumber": "WF-2024-00001",
    "status": "PENDING",
    "currentLevel": 2,
    "nextApprover": {
      "id": "next-approver-uuid",
      "fullName": "Finance Manager"
    }
  }
}
```

Final Approval Response:

[JSON]

```
{
  "success": true,
  "message": "Workflow fully approved",
  "data": {
    "id": "instance-uuid",
    "referenceNumber": "WF-2024-00001",
    "status": "APPROVED",
    "completedAt": "2024-01-16T15:45:00Z"
```

```
  }
}
```

6. Reject Workflow

6.1 Endpoint

[HTTP]

```
POST /api/approvals/{instanceId}/reject
```

6.2 Request Body

[JSON]

```
{
  "comments": "Rejected due to insufficient budget justification. Please resubmit with detailed cost breakdown."
}
```

Request Parameters:

Parameter	Type	Required	Description
comments	string	Yes	Reason for rejection (required)

6.3 Response

[JSON]

```
{
  "success": true,
  "message": "Workflow rejected",
  "data": {
    "id": "instance-uuid",
    "referenceNumber": "WF-2024-00001",
    "status": "REJECTED",
    "rejectedAt": "2024-01-16T14:30:00Z",
    "rejectedBy": {
      "id": "approver-uuid",
      "fullName": "Manager One"
    }
  }
}
```

7. Escalate Workflow

7.1 Endpoint

[HTTP]

```
POST /api/approvals/{instanceId}/escalate
```

7.2 Request Body

[JSON]

```
{
  "comments": "Escalating due to amount exceeding my approval limit.",
  "targetUserId": "target-approver-uuid"
}
```

Request Parameters:

Parameter	Type	Required	Description
comments	string	Yes	Reason for escalation
targetUserId	string	No	Specific user to escalate to (optional)

7.3 Response

[JSON]

```
{
  "success": true,
  "message": "Workflow escalated successfully",
  "data": {
    "id": "instance-uuid",
    "referenceNumber": "WF-2024-00001",
    "status": "ESCALATED",
    "currentLevel": 2,
    "escalatedTo": {
      "id": "next-approver-uuid",
      "fullName": "Finance Director"
    }
  }
}
```

8. Email Approval

Email approval allows approvers to process workflows via links in email notifications without logging into the system.

8.1 Verify Token

[HTTP]

```
POST /api/email-approval/verify
```

Request Body:

[JSON]

```
{
  "token": "email-approval-token-string"
}
```

Response:

```[JSON]
{
  "success": true,
  "data": {
    "valid": true,
    "instanceId": "instance-uuid",
    "referenceNumber": "WF-2024-00001",
    "workflowName": "Purchase Request",
    "action": "APPROVE",
    "expiresAt": "2024-01-22T10:30:00Z",
    "instance": {
      "initiator": "John Doe",
      "amount": 2500.00,
      "summary": "Office supplies request"
    }
  }
}
```

8.2 Process Email Approval

```[HTTP]
POST /api/email-approval/process
```

Request Body:

```[JSON]
{
  "token": "email-approval-token-string",
  "action": "APPROVE",
  "comments": "Approved via email"
}
```

Response:

```[JSON]
{
  "success": true,
  "message": "Workflow approved successfully via email",
  "data": {
    "instanceId": "instance-uuid",
    "referenceNumber": "WF-2024-00001",
    "status": "APPROVED",
    "actionSource": "EMAIL"
  }
}
```

8.3 Token Error Responses

Error	Description
TOKEN_INVALID	Token is malformed or does not exist
TOKEN_EXPIRED	Token has expired
TOKEN_USED	Token has already been used
INSTANCE_ALREADY_PROCESSED	Workflow is no longer in pending state

9. Approval History

9.1 Get Instance History

Approval history is included in the instance detail response. You can also query it separately:

```http
GET /api/workflows/instances/{id}/history
```

9.2 History Entry Structure

```json
{
  "id": "history-uuid",
  "action": "APPROVED",
  "level": 1,
  "actor": {
    "id": "user-uuid",
    "username": "manager1",
    "fullName": "Manager One"
  },
  "actionDate": "2024-01-16T14:30:00Z",
  "comments": "Approved. Within budget allocation.",
  "source": "SYSTEM",
  "ipAddress": "192.168.1.100",
  "userAgent": "Mozilla/5.0..."
}
```

9.3 Action Types

Action	Description
SUBMITTED	Initial submission by initiator
APPROVED	Approved by an approver
REJECTED	Rejected by an approver
ESCALATED	Escalated to higher authority
RECALLED	Recalled by initiator
CANCELLED	Cancelled by initiator or system
RETURNED	Returned to previous level
REASSIGNED	Reassigned to different approver

9.4 Action Source Types

Source	Description
SYSTEM	Action performed via web interface
EMAIL	Action performed via email approval link
API	Action performed via direct API call
SCHEDULER	Action performed by system scheduler (auto-escalation)

10. Error Handling

10.1 Common Errors

Error Code	HTTP Status	Description
INSTANCE_NOT_FOUND	404	Workflow instance does not exist
NOT_ASSIGNED_APPROVER	403	User is not the current approver
INVALID_STATUS	400	Instance is not in approvable state
COMMENT_REQUIRED	422	Comment is required but not provided
ALREADY_PROCESSED	400	Instance already processed
CANNOT_ESCALATE	400	No higher approval level exists

10.2 Error Response Format

[JSON]

```
{
  "success": false,
  "message": "You are not assigned as the approver for this workflow",
  "error": "NOT_ASSIGNED_APPROVER",
  "details": {
    "currentApprover": "manager2",
    "requestingUser": "manager1"
  }
}
```

11. Code Examples

11.1 JavaScript Approval Flow

[JavaScript]

```
class ApprovalService {
  constructor(apiUrl, token) {
    this.apiUrl = apiUrl;
    this.headers = {
      'Authorization': `Bearer ${token}`,
```

```
      'Content-Type': 'application/json'
    };
  }

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

  async getApprovalDetail(instanceId) {
    const response = await
fetch(`${this.apiUrl}/api/approvals/${instanceId}`, {
      headers: this.headers
    });
    return response.json();
  }

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

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

// Usage
const approvalService = new ApprovalService('https://api.server.com',
token);
const pending = await approvalService.getPendingApprovals();
console.log(`${pending.data.totalElements} pending approvals`);

// Approve first one
const result = await approvalService.approve(
```

```
  pending.data.content[0].id,
  'Approved - verified and within budget'
);
```

11.2 Python Approval Example

[Python]

```
import requests

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

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

    def approve(self, instance_id, comments):
        response = requests.post(
            f"{self.base_url}/api/approvals/{instance_id}/approve",
            headers=self.headers,
            json={"comments": comments}
        )
        return response.json()

    def reject(self, instance_id, comments):
        response = requests.post(
            f"{self.base_url}/api/approvals/{instance_id}/reject",
            headers=self.headers,
            json={"comments": comments}
        )
        return response.json()

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

for approval in pending["data"]["content"]:
    print(f"{approval['referenceNumber']}:
{approval['workflow']['name']}")

# Approve
```

```
result = client.approve("instance-uuid", "Approved after review")
print(result)
```

11.3 cURL Examples

[Bash]

```
# List pending approvals
curl -X GET "https://api.server.com/api/approvals" \
  -H "Authorization: Bearer $TOKEN"

# Get approval details
curl -X GET "https://api.server.com/api/approvals/instance-uuid" \
  -H "Authorization: Bearer $TOKEN"

# Approve
curl -X POST "https://api.server.com/api/approvals/instance-
uuid/approve" \
  -H "Authorization: Bearer $TOKEN" \
  -H "Content-Type: application/json" \
  -d '{"comments":"Approved after review"}'

# Reject
curl -X POST "https://api.server.com/api/approvals/instance-uuid/reject"
\
  -H "Authorization: Bearer $TOKEN" \
  -H "Content-Type: application/json" \
  -d '{"comments":"Rejected - insufficient documentation"}'
```