

Task Status Workflow Guide for Frontend

Overview

This guide explains how to move tasks through workflow statuses with a **SINGLE API CALL** that can update both status and task fields simultaneously.

Key Endpoint

PUT `/api/v1/projects/{projectId}/tasks/{taskId}/status`

This endpoint now supports:

-  Status change with workflow validation
-  Optional task field updates (title, description, priority, category, etc.)
-  Comment addition
-  Flag value assignment
-  All in ONE API call

Request Schema

```
{
  "to_status_key": "NEW",           // REQUIRED
  "comment": "string",             // Optional (required if COMMENT_FORMAT requirement)
  "flag_value_ids": ["uuid"],      // Optional (required if FLAGS_REQUIRED requirement)

  // Optional task field updates
  "title": "string",
  "description": "string",
  "priority": "LOW|MEDIUM|HIGH",
  "category_id": "uuid",
  "subcategory_id": "uuid",
  "suite_number": "string",
  "assigned_to": "uuid",
  "start_date": 1234567890000,
  "end_date": 1234567890000
}
```

Complete Workflow Example

Task ID: `f163795d-404c-418b-8b22-df77fa618555`

Project ID: 381a4a60-40c2-4f45-a817-7fc7d84349fd

Step 1: DRAFT → NEW

Requirements: PHOTO_OR_VIDEO, SERIAL_NUMBER_PHOTO, COMMENT_FORMAT

Allowed Roles: ADMIN, PM, SUPERVISOR

Prerequisites:

1. Upload at least 1 photo: `POST /api/v1/projects/{projectId}/tasks/{taskId}/attachments`
2. Upload serial number photo with `is_serial: true`

Request:

```
PUT /api/v1/projects/381a4a60-40c2-4f45-a817-7fc7d84349fd/tasks/f163795d-404c-418b-8b22-df
{
  "to_status_key": "NEW",
  "comment": "John Doe - Task reviewed and approved. Serial: SN-12345-ABC",
  "priority": "HIGH",
  "assigned_to": "user-uuid-here"
}
```

Step 2: NEW → TO_INSTALL

Requirements: PHOTO_OR_VIDEO, SERIAL_NUMBER_PHOTO, COMMENT_FORMAT, CATEGORY_SELECTION, FLAGS_REQUIRED

Flag Group: Appliance Issues (de4aa038-5892-424a-a415-3b1f14de3ef1)

Allowed Roles: ADMIN, SUPERVISOR, PM, VIEWER

Request:

```
PUT /api/v1/projects/381a4a60-40c2-4f45-a817-7fc7d84349fd/tasks/f163795d-404c-418b-8b22-df
{
  "to_status_key": "TO_INSTALL",
  "comment": "Supervisor - Ready for installation. All materials verified.",
  "flag_value_ids": ["521ef234-a6f2-4b20-b614-fbc83ed68dc0"],
  "category_id": "246a42d3-dbd3-461c-9a17-842c73ed1175",
  "suite_number": "Suite 101"
}
```

Available Flags (Appliance Issues):

- 521ef234-a6f2-4b20-b614-fbc83ed68dc0 - Damage Reported
- fb79beca-0e0d-4c37-ae3a-18a602cbdacf - Delivered Damaged
- e5129d6a-a2cf-4544-a38f-4a979b3a2de4 - Missing Parts
- 4543dcb7-af4f-4987-834b-88d70378fba7 - Pending Materials

Step 3: TO_INSTALL → COMPLETED

Requirements: PHOTO_OR_VIDEO, SERIAL_NUMBER_PHOTO, CATEGORY_SELECTION

Allowed Roles: ADMIN, SUPERVISOR

Request:

```
PUT /api/v1/projects/381a4a60-40c2-4f45-a817-7fc7d84349fd/tasks/f163795d-404c-418b-8b22-df
{
  "to_status_key": "COMPLETED",
  "comment": "Installation completed successfully. All checks passed.",
  "description": "Installation completed with all safety checks verified"
}
```

Step 4: COMPLETED → VERIFIED

Requirements: PHOTO_OR_VIDEO, COMMENT_FORMAT, FLAGS_REQUIRED

Flag Group: Trade Issues (0318c140-184f-4b67-b7b5-28bdee58b630)

Allowed Roles: ADMIN, PM, SUPERVISOR

Request:

```
PUT /api/v1/projects/381a4a60-40c2-4f45-a817-7fc7d84349fd/tasks/f163795d-404c-418b-8b22-df
{
  "to_status_key": "VERIFIED",
  "comment": "PM Name - Installation verified by supervisor.",
  "flag_value_ids": ["091da682-58ab-4d2a-a6bd-15dec6c8ecf3"]
}
```

Available Flags (Trade Issues):

- 091da682-58ab-4d2a-a6bd-15dec6c8ecf3 - Electrical Issues
 - a1dd87a2-8f41-4f75-a64d-37310c132cd8 - Cabinet Issues
-

Step 5: VERIFIED → CANNOT_INSTALL_APPLIANCE

Requirements: PHOTO_OR_VIDEO, SERIAL_NUMBER_PHOTO, COMMENT_FORMAT, CATEGORY_SELECTION

Allowed Roles: ADMIN, PM

Request:

```
PUT /api/v1/projects/381a4a60-40c2-4f45-a817-7fc7d84349fd/tasks/f163795d-404c-418b-8b22-df
{
  "to_status_key": "CANNOT_INSTALL_APPLIANCE",
  "comment": "Admin - Appliance defect found during verification. Cannot proceed."
}
```

Step 6: CANNOT_INSTALL_APPLIANCE → CANNOT_INSTALL_TRADE

Requirements: PHOTO_OR_VIDEO, SERIAL_NUMBER_PHOTO

Allowed Roles: ADMIN, PM

Request:

```
PUT /api/v1/projects/381a4a60-40c2-4f45-a817-7fc7d84349fd/tasks/f163795d-404c-418b-8b22-df
{
  "to_status_key": "CANNOT_INSTALL_TRADE",
  "comment": "Trade issue identified - electrical work incomplete."
}
```

Step 7: CANNOT_INSTALL_TRADE → DEFICIENCY

Requirements: PHOTO_OR_VIDEO

Allowed Roles: ADMIN, VIEWER

Request:

```
PUT /api/v1/projects/381a4a60-40c2-4f45-a817-7fc7d84349fd/tasks/f163795d-404c-418b-8b22-df
{
  "to_status_key": "DEFICIENCY",
  "comment": "Deficiency logged. Requires attention from trade team.",
  "priority": "HIGH"
}
```

```
}
```

Step 8: DEFICIENCY → REWORK_NOT_COMPLETED

Requirements: PHOTO_OR_VIDEO

Allowed Roles: ADMIN, PM, SUPERVISOR, INSTALLER

Request:

```
PUT /api/v1/projects/381a4a60-40c2-4f45-a817-7fc7d84349fd/tasks/f163795d-404c-418b-8b22-df
{
  "to_status_key": "REWORK_NOT_COMPLETED",
  "comment": "Rework assigned. Awaiting completion.",
  "assigned_to": "installer-user-uuid"
}
```

Step 9: REWORK_NOT_COMPLETED → REWORK_COMPLETED

Requirements: CATEGORY_SELECTION

Allowed Roles: ADMIN, PM, SUPERVISOR, INSTALLER

Request:

```
PUT /api/v1/projects/381a4a60-40c2-4f45-a817-7fc7d84349fd/tasks/f163795d-404c-418b-8b22-df
{
  "to_status_key": "REWORK_COMPLETED",
  "comment": "Rework completed. All issues resolved.",
  "category_id": "246a42d3-dbd3-461c-9a17-842c73ed1175"
}
```

Frontend Implementation Guidelines

1. Single API Call Pattern

 **DO THIS:**

```
// Update status AND task fields in one call
```

```
await updateTaskStatus(projectId, taskId, {
  to_status_key: "NEW",
  comment: "John - Approved",
  priority: "HIGH",
  assigned_to: userId,
  suite_number: "Suite 101"
});
```

❌ DON'T DO THIS:

```
// Two separate API calls (inefficient)
await updateTask(projectId, taskId, { priority: "HIGH" });
await updateTaskStatus(projectId, taskId, { to_status_key: "NEW" });
```

2. Requirement Validation

Before calling the API, validate requirements on the frontend:

```
const validateRequirements = (status, task) => {
  const requirements = status.requirements;

  if (requirements.includes('PHOTO_OR_VIDEO')) {
    if (task.attachments.length === 0) {
      return { valid: false, message: 'At least one photo/video required' };
    }
  }

  if (requirements.includes('SERIAL_NUMBER_PHOTO')) {
    if (!task.attachments.some(a => a.is_serial)) {
      return { valid: false, message: 'Serial number photo required' };
    }
  }

  if (requirements.includes('COMMENT_FORMAT')) {
    // Validate comment format: "Name - Remark"
    if (!comment || !/^.+\s*-\s*+.+$/.test(comment)) {
      return { valid: false, message: 'Comment must follow format: Name - Remark' };
    }
  }

  if (requirements.includes('CATEGORY_SELECTION')) {
    if (!task.category_id) {
      return { valid: false, message: 'Category selection required' };
    }
  }

  if (requirements.includes('FLAGS_REQUIRED')) {
    if (!flagValueIds || flagValueIds.length === 0) {
      return { valid: false, message: 'Flag values required' };
    }
  }
}
```

```
}

return { valid: true };
};
```

3. Role-Based UI

Show/hide status transition buttons based on user role:

```
const canTransition = (transition, userRole) => {
  return transition.allowedRoles.includes(userRole);
};
```

4. Error Handling

```
try {
  await updateTaskStatus(projectId, taskId, payload);
} catch (error) {
  if (error.code === 'REQUIREMENT_NOT_MET') {
    // Show specific requirement error
    showError(error.message);
  } else if (error.code === 'TRANSITION_NOT_ALLOWED') {
    // User doesn't have permission
    showError('You do not have permission for this transition');
  } else if (error.code === 'INVALID_TRANSITION') {
    // Invalid workflow transition
    showError('This status transition is not allowed');
  }
}
```

Supporting Endpoints

Upload Attachment

```
POST /api/v1/projects/{projectId}/tasks/{taskId}/attachments
Content-Type: multipart/form-data

file: [binary]
is_serial: true/false
```

Get Workflow Configuration

```
GET /api/v1/projects/{projectId}/workflow
```

Get Flag Groups

```
GET /api/v1/flag-groups?company_id={companyId}
```

Get Flag Values

```
GET /api/v1/flag-groups/{groupId}/values
```

Benefits of Single API Approach

1.  **Reduced Network Calls** - One request instead of two
2.  **Atomic Operations** - All changes succeed or fail together
3.  **Better UX** - Faster response time
4.  **Simpler Code** - Less state management
5.  **Consistent Data** - No partial updates

Notes

- **Comment Format:** Must match pattern `Name - Remark` (e.g., "John Doe - Task completed")
- **Flag Values:** Must belong to the flag group mapped to the destination status
- **Category:** Task must have category_id set before transitioning to statuses requiring CATEGORY_SELECTION
- **Attachments:** Upload attachments BEFORE status transition if required
- **Role Permissions:** Backend validates user role against transition allowedRoles