Spec System High Level Design

**Table of Contents**

[1. Purpose / Scope 3](#_Toc110850712)

[2. Reference Documents 3](#_Toc110850713)

[3. Overview 3](#_Toc110850714)

[4. Definitions 3](#_Toc110850715)

[5. Architecture 3](#_Toc110850716)

[6. Security 4](#_Toc110850717)

[7. Setup 5](#_Toc110850718)

[7.2 Approval Matrix Setup 5](#_Toc110850719)

[8. Spec Processing 5](#_Toc110850720)

[8.1 Create new Spec 6](#_Toc110850721)

[8.2 Revise an existing Spec 6](#_Toc110850722)

[8.3 Update spec in Draft state 6](#_Toc110850723)

[8.4 Submit for Approval 7](#_Toc110850724)

[8.5 Review 7](#_Toc110850725)

[8.6 Approved 7](#_Toc110850726)

[9. Schema 7](#_Toc110850727)

[9.1 Role 7](#_Toc110850728)

[9.2 Approval Matrix 8](#_Toc110850729)

[9.3 Spec 9](#_Toc110850730)

[10. API 10](#_Toc110850731)

[10.1 Roles 10](#_Toc110850732)

[10.2 Approval Matrix 11](#_Toc110850733)

# Purpose / Scope

Purpose – This document is the high-level design of the Spec System built to support quality system controls in a light-weight system. It is intended to be part of a company’s Quality Management System (QMS).

Scope – This document is focused on how this Spec System works and how data is stored.

# Reference Documents

N/A

# Overview

Spec System is a light-weight solution for document control. It is a controlled system for releasing, revising and obsoleting documents.

To assure the user interface does not become complicated, this system only supports simple independent specs. A full Product Life Management (PLM) system should be used for tracking spec sets that need to be revised together.

This system is designed to be a stand-alone application. It can use LDAP for getting a list of users and their permissions.

It does have a public API which can be used to perform any tasks that are performed through the user interface.

# Definitions

|  |  |
| --- | --- |
| AD | Microsoft Active Directory |
| API | Application Programming Interface |
| QMS | Quality Management System |

# Architecture

The system is built using a classic three-tier-client-architecture. The system is written in Python using the Django framework. This framework allows the use of different databases including SQL Server and SQLite. The system uses the database to store meta data of the specs. The raw datafiles are stored in the underlying operating system's filesystem. The system is built to enable simple use of Docker to containerize the database and the Python Django webserver. The picture below is an overview of the top-level architecture.



The picture below shows an example implementation where the spec system and the database are implemented using Docker.



# Security

Connection to the spec system will be via https.

Session can be authenticated from one of these methods:

1. OAUTH, for web access
2. Application generated token, for automation access
   1. This token is generated by the application and associates to a service account
   2. A token can be generated by an administrator and expires after one year
   3. A separate token will be generated for each service client connection
   4. A web page allows viewing or removing authorization tokens

Authentication is performed using ldaps to verify the account/password credentials. Ldaps is also used to find AD group membership to provide role-based access.

Roles:

* Administrator – super user with all access – AD Group: SPEC-Admin-Prod (-Test)
* ReadAll – ability to read all confidential specs. – AD Group: SPEC -ReadAll-Prod (-Test)
* User – general SPEC user. Create, Sign, View – AD Group: SPEC-User-Prod (-Test)

Document access approach:

1. Data in the Spec systems are by default classified as Company Propriety and Confidential. (not for public internet)
2. We need to support view by link w/o logging in from within the network (e.g., WI, Policies, etc.)
3. We need to support view by authorized named user(s) only. (e.g.: Restricted data, Trade Secret, etc.)
4. And by default, we should also support read only by any logged-in users. For Information that is not Restricted, but not fully open.  (e.g.: Design spec, System spec, etc.)
   1. i.e.: files that we don't want any employee, contractors, operators to get and take, and has higher value then WI, but not as high Trade Secret, without alerting us of their scrubbing action. (proactively or reactively)
   2. A checkbox will be available for Administrator to set: "Page available w/o login", if document type is not confidential.

# Setup

The following sections describe the setup required before specs can be created and routed. They are listed here to highlight the dependency chain.

* 1. Document Types

Document Types are an organizational way of categorizing documents that serve a type of function. These could be Work Instructions, Change Notices, etc.

Document types have:

* A confidential attribute to indicate they can only be viewed by certain people.
* Jira Template – A link to the Jira story to clone for spec releases of this Approval Matrix
  1. Roles

Roles describe a function the signer is fulfilling or a ‘hat’ the signer is wearing. They can be setup to allow any user to be assigned the role, like Author. Or they can be setup to require one of a list of people to sign, like Document Control Manager.

Roles have:

* A distinct identifier (role). This is the short name of the role. It cannot contain spaces or punctuation
* Description
* Allowed signers - comma separated list of user ids allowed to perform this role. If blank, any user can be specified to fulfill this role
* Specify One – When true, a specific user must be assigned to the role. When false, any user in Allowed Signers can perform the function.

## Departments

Departments on the organizational units that own the documents. The department name is a : separated list of units. So, Operations:Fab1:Zone2:EPIN could be used to describe the Operations department, Fab1 sub department, Zone2 area and EPIN step.

## Approval Matrix

Approval Matrix defines a set of attributes that control how a document is processed and viewed.

An Approval Matrix has:

* Required Roles – A comma separated list of roles that must approve each spec released in this Approval Matrix
* Can Read Roles – A comma separated list of roles that contain people that can read confidential documents.

# Spec Processing

Spec states:



Spec workflow:



## Create new Spec

Click on New button. Select the Approval Matrix, fill in the Title and click Save to create a new spec. It will be given a new unique number and set to revision A.

Continue processing at 8.3 below

## Revise an existing Spec

Locate the spec to be revised. Click on the Revise button. Enter the Reason for Change and click Save. A new revision will be created with the next alphabetical revision (B-Z, AA-ZZ)

Each user that has elected to ‘watch’ this spec is notified via email.

Continue processing at 8.3 below

## Update spec in Draft state

While the spec is in the draft state, all edits must be made. Once submitted for approval, all files and meta-data are locked and cannot be updated.

### Files

Each spec must have at least one file attached to be submitted for signoff. The files should be ordered, so the first file in the list has the header information. On submittal, the files (that can be) will be rendered into one PDF in the order specified.

### Update Jira

The related Jira story contains the Change Impact Assessment. Each of the tasks must be reviewed. For tasks that do not apply to this spec release, enter the reason it does not apply in the task and set the task state to ‘Not Applicable’. For tasks that apply, update the description to reflect that and set the assignee of the task to the person who will perform that task.

### Reviewers

All roles from the Approval Matrix will automatically be copied to this spec and cannot be removed.

Additional roles can be manually added. These manually added roles can be removed, if created in error.

For each role where Specify One is true, a specific user must be identified that will perform this approval.

### References

Update references as needed.

## Submit for Approval

When everything is ready for approval, click Submit button.

This will change the state of the spec to Submitted. In this state no changes can be made to the meta data or the attached files.

A PDF will be generated from the documents attached and added to the spec’s file list.

Email notifications will be sent to all users that were specifically assigned to any role. Email will not be sent to users that are part of a role, but not individually assigned. Email will also to be sent to users who have elected to ‘watch’ this spec.

Modify Jira ticket to sign off to prevent editing during review.

## Review

Each reviewer is responsible for reviewing the spec for accuracy and completeness. The generated PDF must be reviewed to ensure it accurately reflects the spec. The Jira story needs to be reviewed to assure the tasks reflect the correct Change Impact actions.

If there is an issue, a reject reason must be entered and then click Reject. The reject reason will be added as a comment in the Jira story. The spec will be returned to the Draft state.

If everything is correct, click the Sign button.

When someone has signed for each of the specified roles, the spec moves to the Active state. The Jira story moves to the Approved state and all people assigned to tasks under the story are notified their tasks need to be completed. Email will also to be sent to the creator of this version and users who have elected to ‘watch’ this spec.

## Approved

Once the spec has been approved, all the tasks in the Jira story must be completed. Document Control will monitor Stories for specs that are active to assure all deployment tasks have been completed.

# Schema

The sections below cover the different tables used by the system.

The \_hist tables are updated via triggers on insert, update and delete to maintain a history of values.

For tables that have a compound primary key, Django will create an *id* column to be the primary key. This column is not included in the schemas listed below or in the interfaces.

## Document Type

Document Types are an organizational way of categorizing documents that serve a type of function. These could be Work Instructions, Change Notices, etc.



## Role

A role is the function the person reviewing the document is performing. Roles can allow anyone to be assigned to perform the approval per document, or can have a defined list of users to be used.



### Role

Holds the role name, description and a Boolean indicating a specific user must be specified.

### Role\_user

Holds the current list of users that may perform this role.

## Department

Departments on the organizational units that own the documents. The department name is a : separated list of units. So, Operations:Fab1:Zone2:EPIN could be used to describe the Operations department, Fab1 sub department, Zone2 area and EPIN step.



### Dept\_read\_role

Contains the list of users that can read confidential documents in this department.

## Approval Matrix

Approval Matrix defines a set of attributes that control how a document is processed and viewed.



### Apvl\_mt\_sign\_role

Holds the list of roles that must approve any spec in this Approval Matrix.

## Spec

A spec is the unit that is routed. It will have meta data, one or more files, one or more



Spec

Meta data about spec. Includes title, document type, department, state and keywords.

Spec\_sig

Signatures required for this spec. If signed, who signed when.

Spec\_hist

History of changes made to this spec. (Create, Edit, Submit, Sign, Reject, …)

Spec\_file

One or more files that are the content of this spec. If incl\_pdf is set, the file will be rendered into the generated pdf.

Spec\_reference

Zero or more specs this spec references

# API

The list of backend API endpoints are in proj/urls.py.

An OPTIONS call to each of these endpoints will return the HTTP request types allowed and a description of the endpoint. The description will include sample bodies in requests, where appropriate.

Graphical user interface, text, application

Description automatically generated