

Spec System High Level Design

Table of Contents

| | | | |
|-----|--------------------------|---|---|
| 1. | Purpose / Scope..... | 2 | |
| 2. | Reference Documents..... | 2 | |
| 3. | Overview..... | 2 | |
| 4. | Definitions..... | 2 | |
| 5. | Architecture..... | 2 | |
| 6. | Security..... | 3 | |
| 7. | Setup..... | 4 | |
| 7.2 | Roles | | 4 |
| 7.3 | Departments | | 5 |
| 7.4 | Approval Matrix | | 5 |
| 8. | Spec Processing..... | 5 | |
| 9. | Schema..... | 5 | |
| 9.1 | Document Type | | 5 |
| 9.2 | Role | | 6 |
| 9.3 | Department | | 6 |
| 9.4 | Approval Matrix | | 7 |
| 9.5 | Spec | | 7 |
| 10. | API..... | 9 | |

1. 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.

2. Reference Documents

N/A

3. 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.

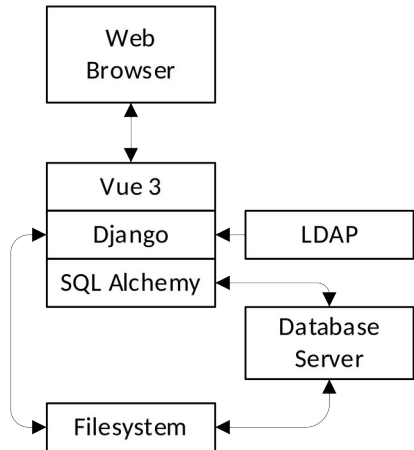
It does have a public API which can be used to perform any tasks that are performed through the user interface.

4. Definitions

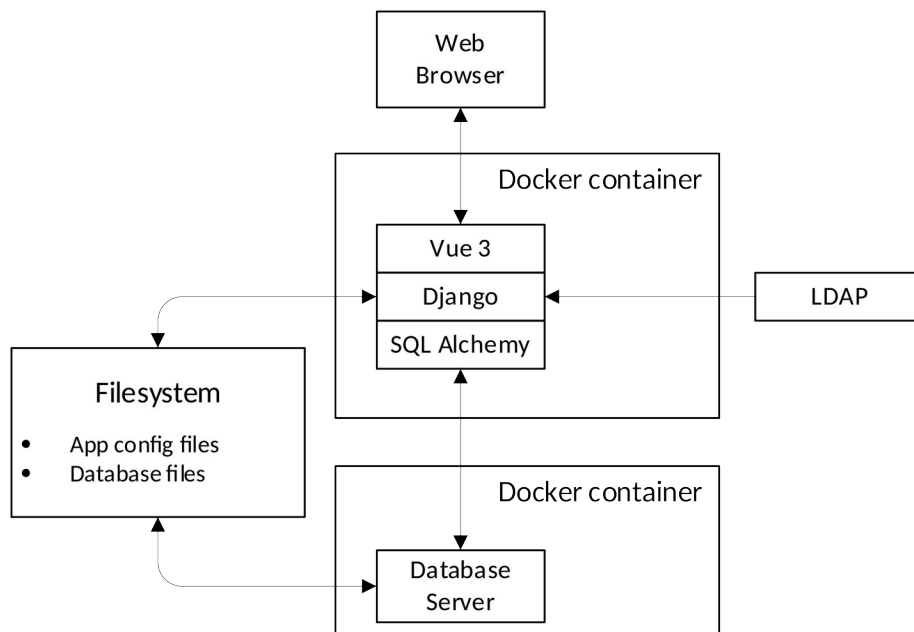
| | |
|-----|-----------------------------------|
| AD | Microsoft Active Directory |
| API | Application Programming Interface |
| QMS | Quality Management System |

5. 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.



6. 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
 - a. This token is generated by the application and associates to a service account
 - b. A token can be generated by an administrator and expires after one year
 - c. A separate token will be generated for each service client connection
 - d. 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.)
 - a. 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)
 - b. A checkbox will be available for Administrator to set: "Page available w/o login", if document type is not confidential.

7. Setup

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

7.1 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 distinct identifier (doc_type). This is the short name of the document type. It cannot contain spaces or punctuation
- Description
- 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 (optional)
- Sunset Interval – Amount of time since activation before spec is automatically obsoleted. (optional) Format: ddd hh:mm:ss (days hours:minutes:seconds)
- Sunset Warning – Amount of time before expiration spec will show up on the sunset list page. (optional)

7.2 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
- Must Specify Signer – When true, a specific user must be assigned to the role. When false, any user in Allowed Signers can perform the function.

7.3 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.

Departments have:

- Read Roles – comma separated list of Roles. The members of these roles can read any confidential document within this department.

7.4 Approval Matrix

Approval Matrix defines a set of attributes that identify what approvals are needed for signoff.

An Approval Matrix has:

- Document Type and Department – Combined identify a unique entry.
 - o The departments are a tree with __Generic__ as the root department. Every Approval Matrix in the tree is applied to a spec.
- Required Roles – A comma separated list of roles that must approve each spec released in this Approval Matrix

8. Spec Processing

Refer to the User Guide.

9. 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.

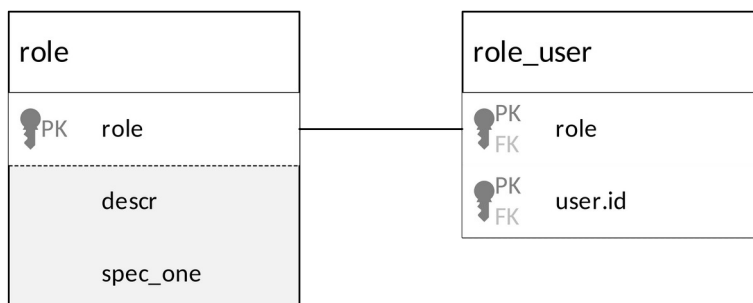
9.1 Document Type

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



9.2 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

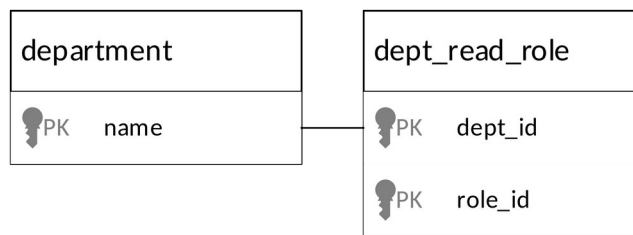
Holds the role name, description and a Boolean indicating a specific user must be specified.

Role_user

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

9.3 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.

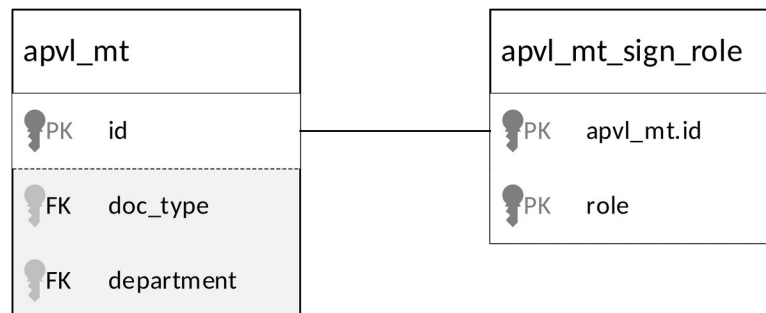


Dept_read_role

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

9.4 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.

9.5 Spec

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

| spec | spec_sig | spec_hist |
|--|--|--|
|  PK num |  PK spec.id |  PK spec.id |
|  PK ver |  PK role.id |  PK mod_ts |
| title |  PK signer_id | upd_by |
| doc_type | delegate_id | change_type |
| department | signed_dt | comment |
| keywords | from_am | |
| state | | |
| created_by | | |
| create_dt | | |
| mod_ts | | |
| jira | | |
| anon_access | | |
| reason | | |
| approved_dt | | |
| sunset_extended_dt | | |

| spec_file | spec_reference |
|--|--|
|  PK spec.id |  PK spec.id |
|  PK filename |  PK num |
| file |  PK ver |
| seq | |
| incl_pdf | |

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

10. 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.

← → ↻ ⚠ Not secure | spec-dev01/role/ 🔗 ☆ ⚙ □ 👤 ⋮

Django REST framework

Role List

Role List

OPTIONS GET ▾

get:
Return list of roles

post:
Create role

```
{  
  "role": "DOC_MGR",  
  "descr": "Document control manager",  
  "spec_one": true,  
  "users": "user1, user2"  
}
```

OPTIONS /role/

HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

```
{  
  "name": "Role List",  
  "description": "get:\nReturn list of roles\npost:\nCreate role\n{\n  \"role\": \"DOC_MGR\",  
  \"spec_one\": true,  
  \"users\": \"user1, user2\"  
}",  
  "renders": [  
    "application/json",  
    "text/html"  
  ],  
  "parses": [  
    "application/json",  
    "application/x-www-form-urlencoded",  
    "multipart/form-data"  
  ]  
}
```