**Software Design Specification**

**Document**

**[Document Template Management System]**

**Version: (1.0.0)** **Date: (01/31/2018)**

**Table of Contents**

1 Introduction 1

1.1 System Overview 1

1.2 Definitions, Acronyms, and Abbreviations 1

1.3 References 1

1.4 Document Map 1

2 Design Considerations 2

2.1 Assumptions 2

2.2 Constraints 2

2.3 System Environment 2

2.4 Design Methodology 2

3 Architectural (High-level) Design 3

3.1 Overview 3

3.2 Rationale 3

3.3 Conceptual (or Logical) View 3

3.4 Other Views 3

4 Low Level Design 4

4.1 Component 1 4

4.2 Component 2 4

4.n Component n 4

5 User Interface Design 5

5.1 Application Control 5

5.2 Screen 1 5

5.3 Screen 2 5

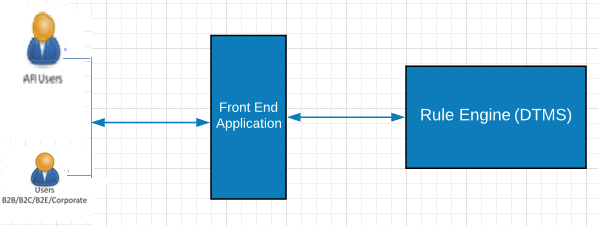
5.n Screen n 5

# Introduction

The following subsections of the Software Design Specifications (SDS) document provides an overview of Document Template Management System.

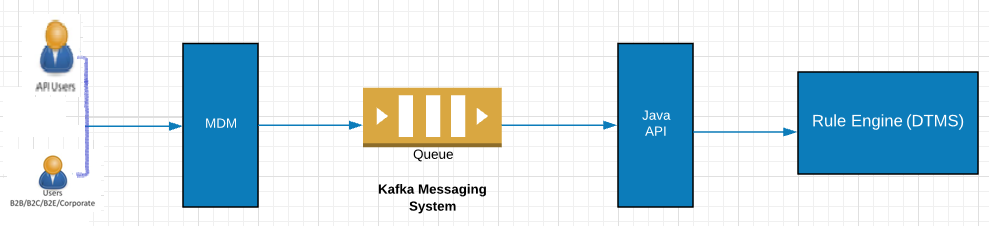
## System Overview

**Consumption Flow:**



1. A user will send a request through a front end system which will then send the request to Document Template Management system.
2. Based on the input conditions, DTMS system will execute the most specific rule that matches the input and determines the appropriate templates to be sent out as output.
3. The templates configured are Email, Physical letter and SMS.
4. These templates will be modified to replace the dynamic fields with their actual values at run time and the updated template will be sent to the consuming application.

**Configuration Flow:**



1. Business users can configure templates from MDM UI.

2. MDM pushes the template content to Kafka messaging layer.

3. DTMS provides a Java API layer which will receive the message from Kafka and transform the template to DTMS format.

4. The Java API will automatically insert the rules into DTM system.

5. This can also be used to automatically update or delete the templates in DTM system.

## Definitions, Acronyms, and Abbreviations

| **Term/Acronym** | **Definition** |
| --- | --- |
| CAPIOT | CAPIOT Software Pvt. Ltd, the proposed vendor of services in this proposal |
| Cnk | Cox and Kings |
| DTMS | Document Template Management System |
| ESB | Enterprise service bus |
| MDM | Master Data Management |
| BE | Booking Engine |
| SI | Supplier Integration |

## References



## Document Map

N/A

# Design Considerations

## Assumptions

2.1.1. All the dynamic fields present in the template will be received as a part of input to the system.

2.1.2. If any dynamic field is not received in the input then those fields will be replaced with a blank space.

## Constraints

BRMS uses JDK 1.8

BRMS uses Apache Ant 1.10.1

## System Environment

* AWS server would be used for deploying web services of DTMS.
* **Software Requirements:**
* Apache Ant 1.10.1
* Java (Jdk 1.8)
* BRMS 6.5
* Eclipse

## Design Methodology

Object oriented programming is used in DTMS. Data objects are created in BRMS Rule Engine to store facts/information. BRMS uses RETE algorithm to decide the rule that needs to be triggered based on various conditions and conflict resolution cycle. The services are exposed as REST services for consumption.

# Architectural (High-level) Design

The architectural patterns followed here are Rule based architecture (using RETE algorithm) and Representational State Transfer (REST).

DTMS service has been exposed as RESTful web services.

## Overview

* This section provides a high level overview of the structural and functional decomposition of DTM system.
* DTMS receives a request from consuming application with various input conditions (like Group of companies, Company, Business Unit, Sub Business Unit, Process, Scenario etc) and dynamic fields (like departure date, employee name etc.).
* It evaluates the input conditions and executes the rule that matches all the input conditions.
* It contains the static templates for Email, Physical letter and SMS formats
* The static templates will be enriched with the dynamic values at runtime and sent out as output.
* If any dynamic value is not sent in the input, then that dynamic value will be replaced with a blank space.
* If there are no matching rules, then "no matching rules" error will be sent to consuming application.

## Rationale

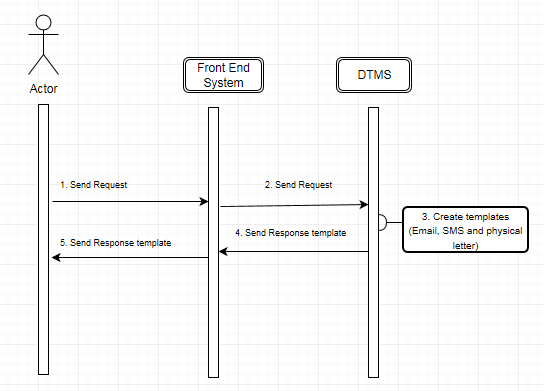
REST architecture has been chosen as REST-compliant Web services allow requesting systems to access and manipulate textual representations of Web resources using a uniform and predefined set of stateless operations. REST systems aim for fast performance, reliability, and the ability to grow, by re-using components that can be managed and updated without affecting the system as a whole.

Rule based architecture is used to derive an action based on certain set of conditions. It includes a set of rules (Rule base), inference engine (that manages evaluation conditions, conflict resolution and actions)

## Conceptual (or Logical) View

## Other Views

3.4.1. Sequence Diagram



# Low Level Design

This section provides the low-level design for each of the system components in DTMS.

# User Interface Design

**N/A**

## Application Control

**N/A**