****

**PROJECT: Business Rule Engine**

 **TECHNICAL SPECIFICATIONS DOCUMENT**

Created: 02/09/2020

**Document Revision History**

| **Release No.** | **Date** | **Revision Description/****Requirement #** | **Modified By**  | **Approved By** | **Approval Date** |
| --- | --- | --- | --- | --- | --- |
| V1.0 | 02/09/2020 | Initial TSD -  | Francisco Guzman |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Table of Contents**

[**1.0**](#_heading=h.30j0zll) **GENERAL INFORMATION 4**

[*1.1*](#_heading=h.1fob9te) *Project Objective 4*

[1.1.1](#_heading=h.3znysh7) Overview 4

[*1.2*](#_heading=h.2et92p0) *System requirements (Tools) 4*

[1.2.1](#_heading=h.tyjcwt) Tools Setup 4

[**2.0**](#_heading=h.3dy6vkm) **Architecture 6**

[*2.1*](#_heading=h.1t3h5sf) *Topology 6*

[*2.2*](#_heading=h.4d34og8) *Systems Architecture 7*

[2.2.1](#_heading=h.2s8eyo1) High level. 7

[2.2.2](#_heading=h.17dp8vu) High-Level Sequence and Trigger Point 8

[*2.3*](#_heading=h.3rdcrjn) *Application Architecture 8*

[2.3.1](#_heading=h.26in1rg) High level design. 8

[2.3.2](#_heading=h.lnxbz9) Document Types. 8

[2.3.3](#_heading=h.35nkun2) Interfaces. 8

[2.3.4](#_heading=h.1ksv4uv) Access Control. 9

[2.3.5](#_heading=h.44sinio) Search (common functionality in all document types). 9

[2.3.6](#_heading=h.2jxsxqh) EFM Document Types and Component Functionality: 9

[2.3.7](#_heading=h.1ci93xb) Current VA Functionality. 11

[2.3.8](#_heading=h.3whwml4) Design to Configure Services 11

[2.3.9](#_heading=h.2bn6wsx) Design to Retrieve Lines of Businesses 12

[*2.4*](#_heading=h.qsh70q) *Technical Design 13*

[**3.0**](#_heading=h.1pxezwc) **INTEGRATION POINTS 13**

[*3.1*](#_heading=h.49x2ik5) *NGV 13*

[**4.0**](#_heading=h.2p2csry) **Impact areas 14**

[*4.1*](#_heading=h.147n2zr) *Document Viewing 14*

[*4.2*](#_heading=h.3o7alnk) *Appointments 14*

[*4.3*](#_heading=h.23ckvvd) *Timeliness Adjustment logic – no effect. 14*

[*4.4*](#_heading=h.ihv636) *Others 14*

[**5.0**](#_heading=h.32hioqz) **Impacted applications 14**

[*5.1*](#_heading=h.1hmsyys) *eProcess - VA EFM 14*

[*5.2*](#_heading=h.41mghml) *RHRP Intake 14*

[**6.0**](#_heading=h.2grqrue) **Database 15**

[*6.1*](#_heading=h.vx1227) *Database Objects 15*

[6.1.1](#_heading=h.3fwokq0) Core Tables. 15

[6.1.2](#_heading=h.1v1yuxt) Table Details. 15

[*6.2*](#_heading=h.4f1mdlm) *ER Diagram 15*

[**7.0**](#_heading=h.2u6wntf) **Appendix 17**

[**8.0**](#_heading=h.19c6y18) **Notes 18**

[**9.0**](#_heading=h.3tbugp1) **References 19**

[**10.0**](#_heading=h.28h4qwu) **Glossary 20**

# GENERAL INFORMATION

## Introduction

The objective of the Business Rule Engine (BRE) is to receive a request with the name of a rule to execute that will yield a certain outcome by recursively making decisions. The advantage to such an engine is that there would no longer be a need to manually perform tasks such as classification, this is especially helpful if a user wants to classify a large dataset. In addition, the engine will be able to offer more functionality than is offered by common operators such as “==”, “>=”, and “<=”. Another great benefit is that if a user wants to add a new rule, the engine won’t require several if statements to accommodate the change. Eventually, after reaching a base case the BRE will return an output based on the rule selected.

###  Purpose and Objective

Because QTC is one of the largest providers of disability and occupational health examination services, thousands of new patients’ records need to be sorted/classified to get the best healthcare possible, this is in addition to categorizing first time patients’ information. In addition to consolidating and managing data, the rule engine is intended to receive necessary inputs of data from data models to execute endpoints. This document will comprehend the data model and the outcome of the endpoint. The Business Rule engine will consist of accomplish the following:

* Access Expression’s, Rule’s, and Parameters from a database
* Recursively execute call until an outcome is determined
* Take in a rule name and a JSON object with several values to evaluate
* Return status of execution and JSON object with data collected

## System requirements (Tools)

This Business Rule Engine will be developed using ASP.NET, C#, and will utilize JSON, and SQL for the backend.

### Tools Setup

1. SQL Server 2019
2. .NET 6.0
3. Oracle for enterprise printing integration
4. SQL Server for case management integration

# Overall Description

##  Product Perspective

The Business Rule Engine will interact with SQL procedure and the front-end dashboard via plugin and API to retrieve necessary data for the data model to execute the outcome for the endpoint. The Business Rule Engine is dependent on authentication and the authentication is responsible for the user’s access level and lines of business that will enable the plugin configuration.

 

*Figure 1: Overall architecture*

##  Systems Architecture

### High level.

The system will incorporate a variety of components such as a database, plugin(file system), and a method for authentication. The database will contain all the required class members(Expression, Rule, Parameter) that can be called at any time to use in the BRE(Business Rule Engine). These respective objects will be injected as a service at runtime as needed. Additionally, the assembly plugin will extend the business logic for common features. These objects will be referenced by the BRE via dependency injection.

**

*Figure 2: Database Model*

### High-Level Sequence

The system will be receiving the “ExecuteRules” request which will contain the name of the rules that are to be executed. The Engine will then execute the rule accordingly and determine the appropriate outcome with the processes it goes through. After which, either the result will be returned to the user or another rule will be executed recursively depending on the outcome.

## Application Architecture

### High level design.



*Figure 3: ExecuteRule Model*

### Document Types.

Inputs and outputs will be a A JSON object of name/value pairs representing the input data required by the expression tree that will be executed by the rule.

### Interfaces.

* + This is a standalone implementation that will help support business decisions.
	+ Starting with our UI that will display user and system errors separately
	+ The implementation of the BRE will be held in the background and will deliver simple decisions.
	+ These decisions will be handled in the Rule Execution process in which we will recursively execute rules until we reach our final output.
	+ Depending on the JSON object passed into our rule expression this will determine a “NegativeAction” or “PositiveAction” from the engine.
		- NegativeAction: Action associated with the rule if expression evaluates negatively
		- PositiveAction: Action associated with the rule if expression evaluates positively

### Access Control.

* Has a singular object which will be our Business Rule Engine that contains our business rule expressions.
* Rule expressions will have access to a database of rules. Using JSON objects passed into the rule expression, our expression will extract rules from a database and compare our rules to the passed JSON object.

## Technical Design



*Figure 4: Technical Design*

# Database

The BRE will pull from a SQL database that holds the several defined rules and expressions.

## Database Objects

### Core Tables.

 TABLE [BRE].[Id]

 TABLE [BRE].[Name]

 TABLE [BRE].[ExpressionId]

 TABLE [BRE].[PositiveAction]

 TABLE [BRE].[PositiveValue]

 TABLE [BRE].[NegativeAction]

 TABLE [BRE].[NegativeValue]

 TABLE [BRE].[LeftOperandType]

 TABLE [BRE].[LeftOperandValue]

 TABLE [BRE].[RightOperandType]

 TABLE [BRE].[RightOperandValue]

 TABLE [BRE].[Operator]

### Table Details.

* The Rule table is connected with the Expression table and the rule takes precedence over an expression, meaning a rule cannot be executed without having a reference to an expression. An expression can be called by several different rules but there can only be one expression per. It is also important to note that “ExpressionId” under the Rule table is a foreign key with Id being the primary key under the Expression table.

## ER Diagram



*Figure 5: ER Diagram*

# Appendix

* Change affects VA, RHRP and QTC users.
* Users part of AD will have access to the site.

# References

* Exam File Manager’s (EFM) Technical Specification Document

# Glossary

**A. ACRONYMS**

**BRE** Business Rule Engine

**JSON** JavaScript Object Notation

**SQL** Structured Query Language

**B. DATA DICTIONARY**