|  | | |
| --- | --- | --- |
|  |  | |
|  |  | **Services and Messaging Design Document** | |
|  |  | Project: SGR- Small Group Renewals | |
|  | |

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Description | Author |
| 22 Mar 2018 | 1.0 | Services and Messaging - Design Document Draft Version | Infosys |
| 04 Apr 2018 | 1.1 | Services and Messaging - Design Document Draft Version | Infosys |
| 09 Apr 2018 | 1.2 | Services and Messaging - Design Document Draft Version | Infosys |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[1. Executive Summary 5](#_Toc511228882)

[1.1 Objective 5](#_Toc511228883)

[1.2 Logical Design 6](#_Toc511228884)

[1.3 Physical Deployment 8](#_Toc511228885)

[1.4 Security 9](#_Toc511228886)

[1.5 Non Functional Requirements 9](#_Toc511228887)

[1.6 Database Schema 10](#_Toc511228888)

[1.7 Server Side and Micro Service Design 10](#_Toc511228889)

[1.7.1 APIs 10](#_Toc511228890)

[1.7.2 Web App and Services Interaction 12](#_Toc511228891)

[1.7.3 Service Implementation details 14](#_Toc511228892)

[1.7.4 Service Design Patterns 15](#_Toc511228893)

[1.7.5 Server side data caching 15](#_Toc511228894)

[1.7.6 Application Dynamic Configuration Management 16](#_Toc511228895)

[1.7.7 Application Monitoring 16](#_Toc511228896)

[1.7.8 Transaction Design 16](#_Toc511228897)

[1.7.9 Integration Flow 16](#_Toc511228898)

[1.7.10 User Stories 17](#_Toc511228899)

[2. Known issues/limitations 30](#_Toc511228900)

Appendices

[Appendix 1. Reference Documents 42](#_Toc464848557)

# Executive Summary

## Objective

The objective of this document is to explain the design of Services and Messaging module for the Business case –Initiate renewals flow. Here the End User I.e. Actuaries, Underwriters would be triggering the Renewals process from the UI for the eligible customers through several steps that happens at backend. These steps are captured as User stories in the document. The User stories are Get Open Queue Details, Get My Queue Details, Load Initiate Criteria, Initiate Renewals.

This document includes details about the Logical design of the system, Deployment details, Security measures, Performance, Services Interaction, Caching mechanism, Configuration management, Transaction and Integration Flow.

## Logical Design

Context Diagram

The Component Diagram in the below link gives a detailed view of the SGR system and the various components involved in it.

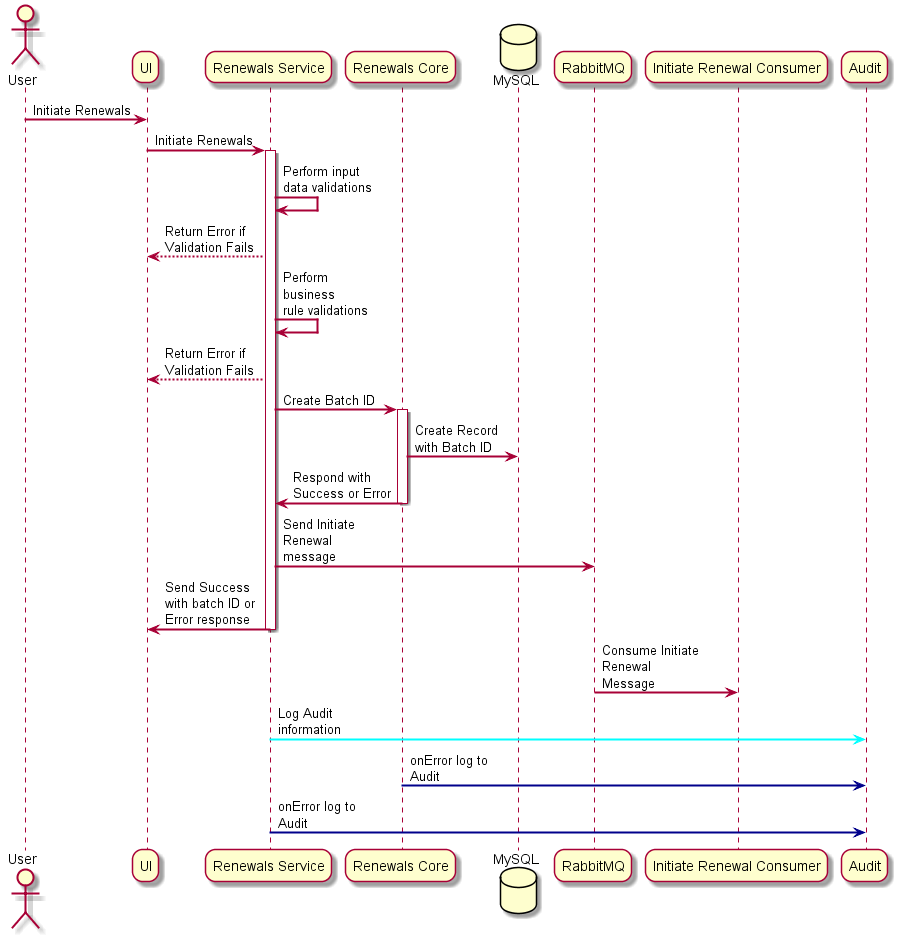
**Renewals High Level Components Diagram:**

Refer to the below link for High Level Component Diagram.

<https://myfyi.sharepoint.com/teams/GrpSales/_layouts/15/DocIdRedir.aspx?ID=GRPSALES-1049839664-44>

Implementation Diagrams

The Implementation diagram gives a pictorial representation of Initial renewals flow. Here the end user (Actuaries/Underwriters) tries to Initiate the renewals process from BCBS by connecting with SGR. The Sequence and flow and the steps are covered briefly below.



***Step1:*** *From BCBS System End user connects with SGR UI through a secure gateway and provides form field data.*

***Step 2:*** *From User Interface a rest service call is done for data and business rule validations.*

***Step 3:*** *Corresponding Error message is sent back to UI in case of failure.*

***Step 4:*** *On Success Renewals core service is called to create a Batch ID.*

***Step 5:*** *Renewals core Service Connects with SQL DB and creates a record for the Batch id created.*

***Step 6:*** *Renewals core service sends a Success Response to Renewals Service.*

***Step 7:*** *Further which Initiate Renewals message is sent to Rabbit MQ which is consumed by Renewals consumer.*

***Step 8:*** *Renewals Service sends Success or Failure message to UI.*

***Step 9:*** *For all of the above transactions both the success and error case logs are captured by the Audit Service.*

***Note:*** *In the above diagram DB should be perceived as* ***MS SQL Server****.*

## Physical Deployment

The Physical Deployment is planned to be on the PCF (PIVOTAL CLOUD FOUNDRY).

The dependent software’s for running the application are given below:

* **MS SQL Server** - Microsoft SQL Server is a relational database management system, or RDBMS, that supports a wide variety of transaction processing, business intelligence and analytics applications in corporate IT environments. It's one of the three market-leading database technologies, along with Oracle Database and IBM's DB2.
* **MongoDB** - MongoDB is a free and open-source distributed database, stores data in flexible, JSON-like documents.
* **Circuit breaker dashboard** - Circuit Breaker Dashboard provides Spring applications with an implementation of the Circuit Breaker pattern. Circuit Breaker Dashboard is based on Hystrix.
* **Spring Boot –** Spring boot is a pre-configured, pre-sugared set of frameworks/technologies to reduce boiler plate configuration providing you the shortest way to have a Spring web application up and running with smallest line of code/configuration out-of-the-box.

* **REDIS** - Redis is an open source (BSD licensed), in-memory **data structure store**, used as a database, cache and message broker. It supports data structures such as [strings](https://redis.io/topics/data-types-intro#strings), [hashes](https://redis.io/topics/data-types-intro#hashes), [lists](https://redis.io/topics/data-types-intro#lists), [sets](https://redis.io/topics/data-types-intro#sets), [sorted sets](https://redis.io/topics/data-types-intro#sorted-sets) with range queries, [bitmaps](https://redis.io/topics/data-types-intro#bitmaps), hyperloglogs and geospatial indexes with radius queries.
* **Service Discovery** - Service discovery is the automatic detection of devices and services offered by these devices on a computer network. A service discovery protocol (SDP) is a network protocol that helps accomplish service discovery.

## Security

**Authentication**

LDAP Authentication:

**Inputs:** Lan ID and Password

On Success

**JWT Authentication**

**Inputs:** User name and Password from Property file.

On Success

Token gets Generation

This token is appended to the headers and a rest service call is done.

## Non Functional Requirements

|  |  |  |
| --- | --- | --- |
| S. No | NFR Component | Details |
| 1 | Peak load | TBD(Need inputs from Geetha on these items) |
| 2 | Average Load | TBD(Need inputs from Geetha on these items) |
| 3 | Service SLA | TBD(Need inputs from Geetha on these items) |
| 4 | Time out definitions for each external service calls | TBD(Need inputs from Geetha on these items) |
| 5 | Define circuit breakers with fall back method for each service module. Come up with parameters for each. | TBD(Need inputs from Geetha on these items) |
| 6 | Identify queue depth for each Queue . | TBD(Need inputs from Geetha on these items) |
| 7 | Define number of consumers for Peak load and Average load | TBD(Need inputs from Geetha on these items) |
| 8 | No of concurrent users logged in  - UI - Peak | TBD(Need inputs from Geetha on these items) |
| 9 | Define Message expiration | TBD(Need inputs from Geetha on these items) |

## Database Schema

TBD

## Server Side and Micro Service Design

### APIs

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Endpoints | Request Params | Response | HTTP Response Codes | Components involved | Description |
| /services/batch/initiate | effectiveDate(String) | Success: Object holding XML/ JSON as response | 200: Success | REST API, MySQL, Java 8, Spring Integration, Redis, RabbitMQ | Initiates an Individual or batch renewals |
|  | Division(string), | Error: Object holding error code with error description as response | 400: Invalid Request |  |  |
|  | marketSegment(string), fundingType(string) |  | 403: Forbidden |  |  |
|  | accountNumber(string), |  | 500: Internal Server Error |  |  |
|  | renewalType(string), |  |  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Endpoints | Request Params | Response | HTTP Response Codes | Components involved | Description |
| /services/queue/openQueue | User Id (String)and User role(String) | Success:Queue Response object containing renewal account information | 200: Success | REST API, Java 8, Spring Integration, | Gets accounts that are not assigned to any user. |
|  |  | Error: Object holding error code with error description as response | 400: Invalid Request |  |  |
|  |  |  | 403: Forbidden |  |  |
|  |  |  | 500: Internal Server Error |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Endpoints | Request Params | Response | HTTP Response Codes | Components involved | Description |
| /services/queue/myQueue | User Id (String) | Success:Queue Response object containing renewal account information | 200: Success | REST API, Java 8, Spring Integration, | Gets accounts that are assigned to any user. |
|  |  | Error: Object holding error code with error description as response | 400: Invalid Request |  |  |
|  |  |  | 403: Forbidden |  |  |
|  |  |  | 500: Internal Server Error |  |  |

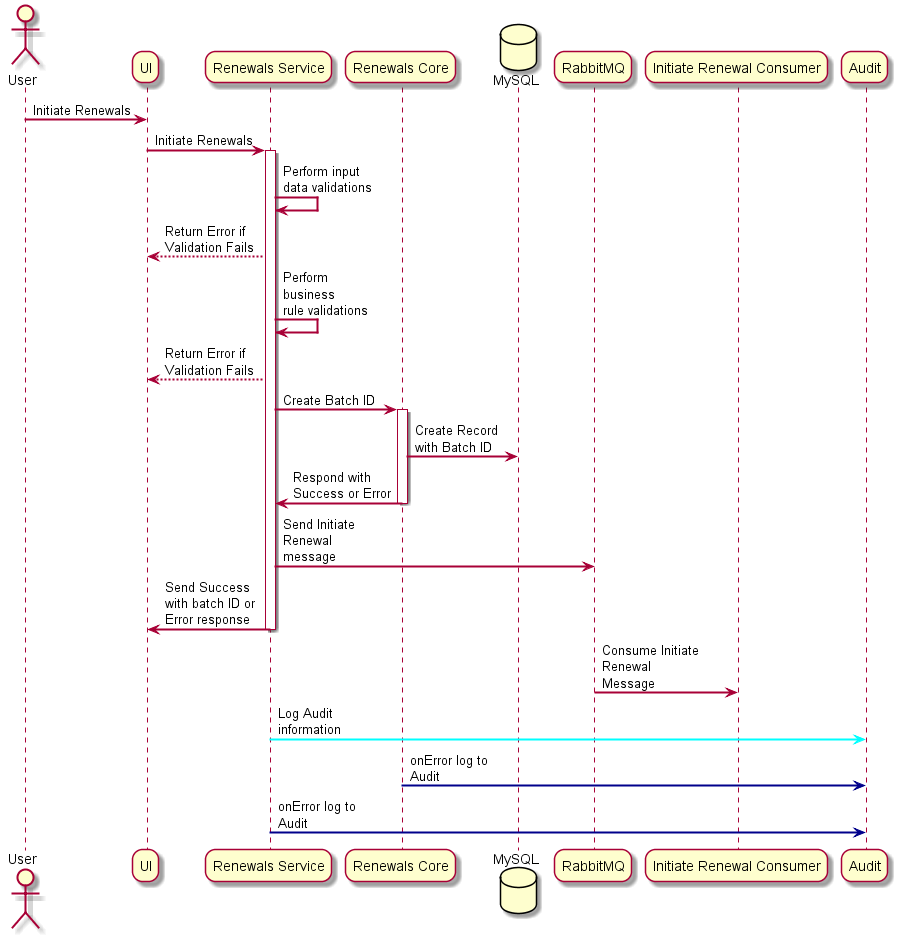
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Endpoints | Request Params | Response | HTTP Response Codes | Components involved | Description |
| /service/batch/criteria | effectiveDate(String) | Success: Object holding XML/ JSON as response | 200: Success | REST API, MySQL, Java 8, Spring Integration, Redis. | Load the Batch Criteria |
|  | Division(string) | Error: Object holding error code with error description as response | 400: Invalid Request |  |  |
|  | marketSegment(string), fundingType(string) |  | 403: Forbidden |  |  |
|  | accountNumber(string) |  | 500: Internal Server Error |  |  |
|  | renewalType(string) |  |  |  |  |

### Web App and Services Interaction

**Description:** Users interact with Rich UI based Presentation Tier which further communicates with the Batch Interface, Business Layer, Persistence Layer and Database.

All the cross cutting concerns like Authentication, Logging, Caching, monitoring etc., are isolated and used across tiers.

The below diagram gives a pictorial representation of Initial renewals flow. Here the end user (Actuaries/Underwriters) tries to Initiate the renewals process from BCBS by connecting with SGR. The Sequence and flow and the steps are covered briefly below.



***Step1:*** *From BCBS System End user connects with SGR UI through a secure gateway and provides form field data.*

***Step 2:*** *From User Interface a rest service call is done for data and business rule validations.*

***Step 3:*** *Corresponding Error message is sent back to UI in case of failure.*

***Step 4:*** *On Success Renewals core service is called to create a Batch ID.*

***Step 5:*** *Renewals core Service Connects with SQL DB and creates a record for the Batch id created.*

***Step 6:*** *Renewals core service sends a Success Response to Renewals Service.*

***Step 7:*** *Further which Initiate Renewals message is sent to Rabbit MQ which is consumed by Renewals consumer.*

***Step 8:*** *Renewals Service sends Success or Failure message to UI.*

***Step 9:*** *For all of the above transactions both the success and error case logs are captured by the Audit Service.*

***Note:*** *In the above diagram DB should be perceived as MS SQL Server.*

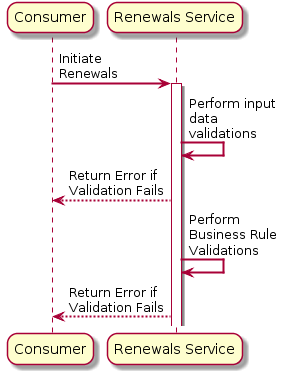
### Service Implementation details

**Renewals Service and Renewals Core Implementation details:**

**Renewals Service – Implementation Details:**

The below diagram explains the interaction between the user and Renewals service module. User’s data Interacts with Renewals service for Input Data and Business Rule validations.

The interactions are explained briefly in the below steps.



**Step 1:** The user initiates renewals by sending a request to the Renewals Service module with the input form field data.

**Step 2:** The input form field data are validated and a response is send back in case of errors.

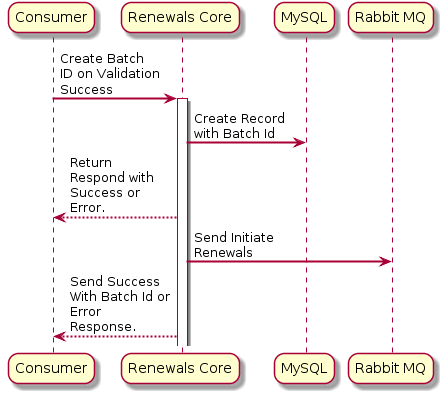
**Step 3:** In case of validation success the business level validations are done and response is send back to the user in case of error.

**Renewals Core – Implementation Details:**

The below diagram explains the interaction between the User - Renewals core module, Renewals core – MS SQL Server and Renewals core - Rabbit MQ. A Batch ID creation request is triggered for Users valid data to Core and a record gets created in MS SQL Server with Batch ID and the renewals process gets initiated in Rabbit MQ.

The Success or failure for both Batch id creation and Renewals initiation process are intimated to the user as response from the corresponding modules.

The interactions are explained briefly in the below steps.

****

**Step 1:** *Renewals core Service Connects with SQL DB and creates a record for the Batch id created, it sends initiate renewals message to Rabbit MQ.*

***Step 2****: Core Responds User with Success or Error on Batch Id creation and Initiating Renewals Process.*

**Note:** In the above diagram DB should be perceived as MS SQL Server.

### Service Design Patterns

TBD

### Server side data caching

**REDIS:** is used for Server side Data Caching

**Redis config:** contains details like Idle timeout, Sequence port etc.,

**Annotations**: @Cacheget, @CachePut, @CacheEvict are used from a project specific jar added as dependency.

**ReddisonClient.jar** is used as client for REDIS.

Annotations are implemented and tested for cache mechanism at POC level.

### Application Dynamic Configuration Management

**Configuration Storage:** Local GIT repository – Application - <Region/Environment.properties>, Application.yml file in local workspace (Since on POC level)

But in the final product delivered, the complete properties and dependencies will be configured in the Application Configuration server in PCF.

Configuration server facilitates configuration from GIT repository and servers per environment.

### Application Monitoring

New Relic will be used for Application Monitoring.

Alerts – TBD.

### Transaction Design

TBD

### Integration Flow

**Spring Integration Flow –DSL (Domain Specific Language).**

**DSL:** If you've ever written or designed a Web page with CSS, you've already encountered a DSL, or domain-specific language. DSLs are small, expressive programming languages custom designed for specific tasks.

HTML is a DSL for Web application development, ant for build etc.,

Spring Integration is used to communicate from one application to the other and also to orchestrate independent services.

**Few Terminologies used in Spring Integration DSL:**

**Gateway:** is Unidirectional Producer sends and Consumer receives the message.

**Adapter:** is bidirectional both producer and consumer can send and receive messages.

**Service Activator:** is similar to interceptor.

**Router:** is similar to if else conditional statements to do logical decisions.

Other Self-explanatory items are **Splitter, Transformer, Aggregator and Filter.**

**Annotations Used:**

@Enable Integration @ Integration component @Gateway @ Service Activator @ Repository etc.,

**Integration Flow:**

**Step 1:** Request gets routed to controller

**Step 2:** Controller to Interface

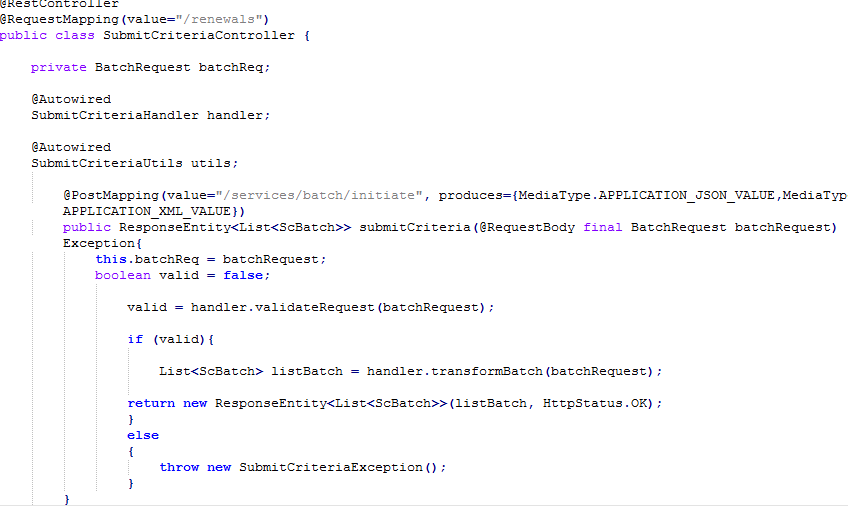
**Step 3:** Interface to Integration flow

The above steps are described briefly in the below sample code and User story.

**Note: The below code and steps are meant for understanding purpose only and not to be associated with any user stories .**

**User story:** Integration flow for initiate renewals

**Step 1:** Submit criteria controller would get called based on the request URL pattern**:/services/batch/initiate**



**Step 2: for this line of code**

**handler.transformBatch(batchRequest)** //normal java method call

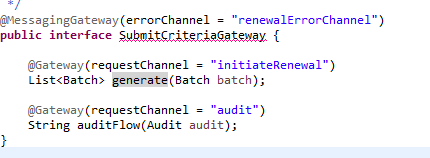
From transformBatch Method refer to code --ProduceMessage(batch) which further calls the gateway to initiate the integration flow,



**Step 3:**

**Refer to code below Gateway.generate(batch) would call the generate method refer to code (List<Batch>generate(Batch batch)) In interface SubmitCriteriaGateway**

cid:image006.png@01D3D1BC.27C44F70



**Step 4: (Actual Integration Flow starts here)**

**Based on the request channel in step 3 refer code //@Gateway(requestChannel = “initiateRenewal”)**

**Integration flow gets executed refer to code below from(initiateRenewal)**



### User Stories

**US #-14415 – Get open queue details**

**Summary:**

To fetch renewal accounts that are unassigned to any particular user.

This service will provide an integration point on the section of the landing page between service module and UI where the renewal accounts that are not assigned to any particular user are displayed based on user id and user role with workflow management section.

**Description:**

Renewals Service – Get open queue details

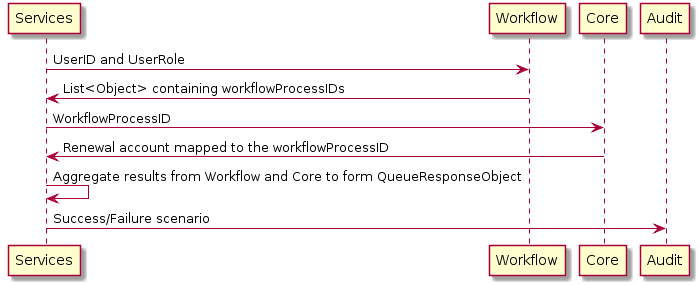
**Assumptions:**

Any business rule based business flow operations are handled in a separate user story. E.g. Transfer/Claim/Assigning of Renewal accounts to users.

|  |  |  |  |
| --- | --- | --- | --- |
| **S No** | **User Story Title** | **User Story Description** | **Notes** |
| 1. | US# 14415 | Renewals Service – Get open queue details | 1. Get workflow details from sprnl-workflow. 2. With workflowProcessId, fetch account information from sprnl-core. 3. Aggregate responses from workflow and core. 4. Send the response to UI |

**Sequence diagram for Get open queue details:**

The below diagram explains the interaction between the Services module with Workflow, Core and Audit. This diagram briefs about how the renewal accounts that are unassigned to any particular user is fetched. The interactions are explained briefly in the below steps.



**Step 1:** User Id and User role are sent as request parameters to workflow module.

**Step 2:** List of workflow process ids is sent back as response.

**Step 3:** This workflow process id is sent to core as request.

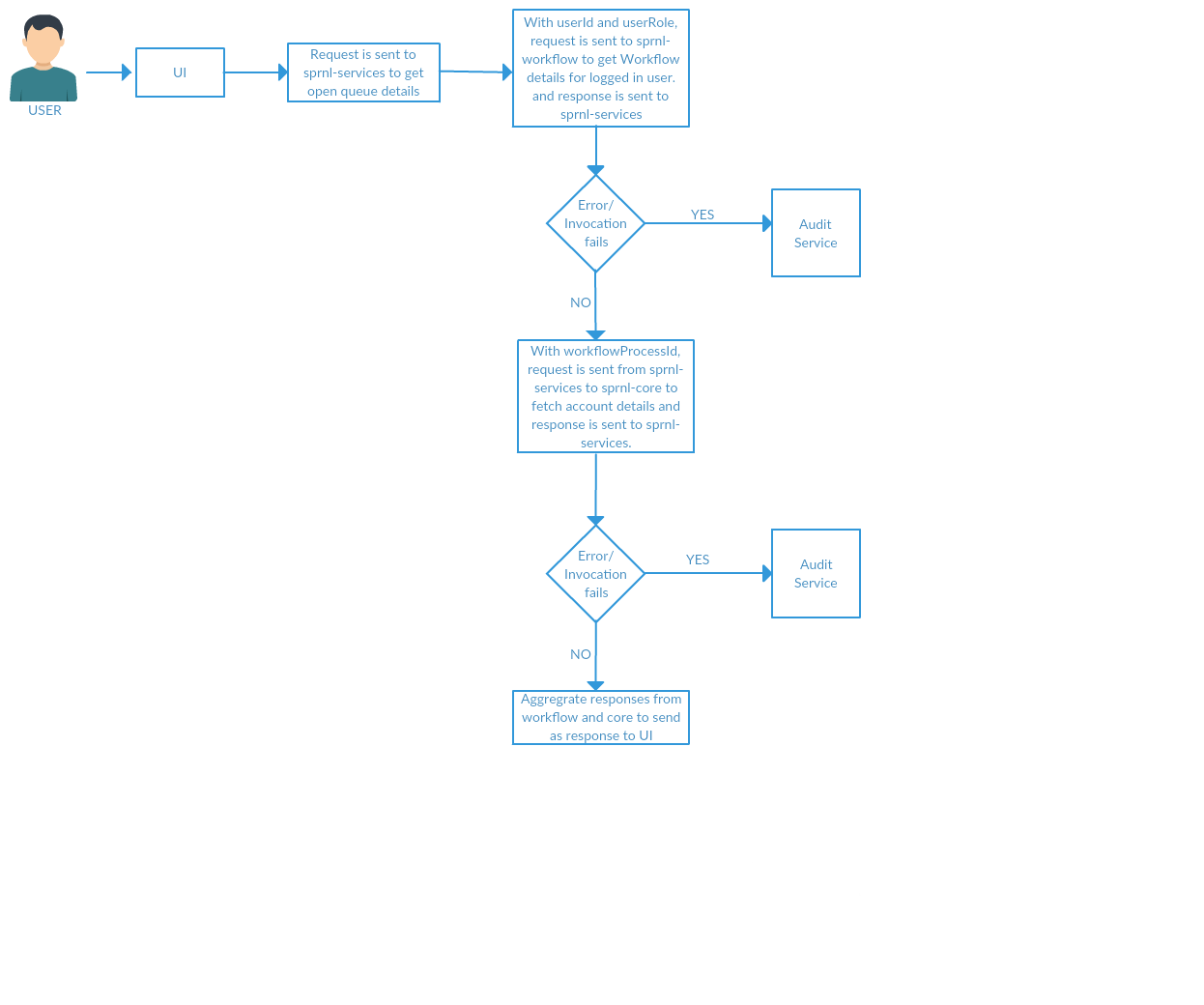
**Step 4:** Renewal accounts mapped to the workflow process id is sent back as response.

**Step 5:** The aggregated results from workflow and core forms the queue response object.

**Step 6:** Success / Failure scenarios are logged in using Audit module.

**Flowchart for Get open queue details:**

The below flow chart explains the interaction between the Services module with Workflow, Core and Audit. This flow chart briefs about how the renewal accounts that are unassigned to any particular user is fetched. The flow of request from one module to the other and the decisions based on responses are covered.



**US #-14417 – Get my queue details**

**Summary:**

This service will provide an integration point between service module and UI for the section on landing page where the renewal accounts which are assigned to the logged in user are displayed based on the user id with workflow management system.

**Description:**

Renewals Service Calls Renewals workflow to get a list of workflow objects which contains WorkflowProcessIDs which is then sent to Renewals Core to get Account Information. The response received from Workflow and Core are aggregated to form the Queue Details response.

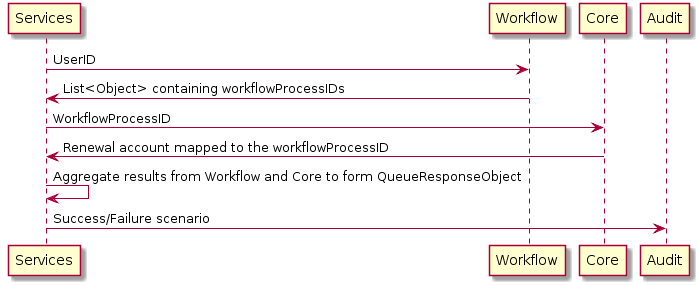
**Assumptions:**

Any Business Rule based business process flow operations are handled in a separate user story e.g. Transfer/Claim/Assigning of Renewals Accounts to user.

|  |  |  |  |
| --- | --- | --- | --- |
| **S. No** | **User Story Title** | **User Story Description** | **Notes** |
| 1 | QRE-14417 – Get My Queue details | Get renewal accounts that are assigned to any particular login user. |  |

**Sequence diagram for Get my queue details:**

The below diagram explains the interaction between the Services module with Workflow, Core and Audit. This diagram briefs about how the renewal accounts that are assigned to any particular user is fetched. The interactions are explained briefly in the below steps.



**Step 1:** User Id is sent as request parameters to workflow module.

**Step 2:** List of workflow process ids is sent back as response.

**Step 3:** This workflow process id is sent to core as request.

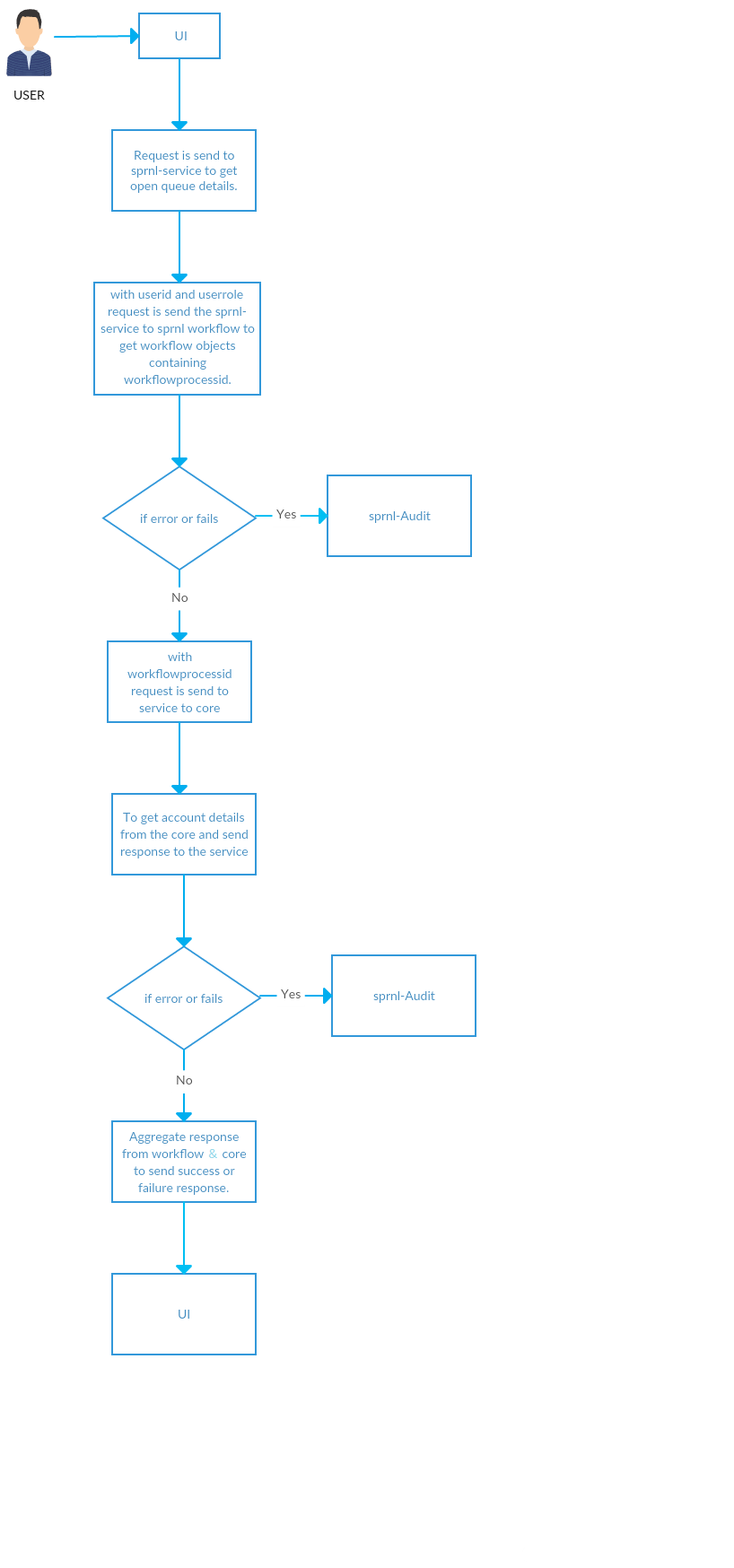
**Step 4:** Renewal accounts mapped to the workflow process id is sent back as response.

**Step 5:** The aggregated results from workflow and core forms the queue response object.

**Step 6:** Success / Failure scenarios are logged in using Audit module.

**Flowchart for Get my queue details:**

The below flow chart explains the interaction between the Services module with Workflow, Core and Audit. This flow chart briefs about how the renewal accounts that are assigned to any particular user is fetched. The flow of request from one module to the other and the decisions based on responses are covered.

****

**US #14310 Initiate Renewal – Load Initiate Criteria**

**Summary:**

To retrieve dropdown values for UI from Redis cache or Database through Service and Core micro services.

This service internally calls a criteria micro service to retrieve all the static [criteria](#Criteria_Glossary) from the database. The criteria response will provide all the obtained criteria available. In case of partial or complete failure in retrieval, the group specific error will be returned along with the obtained criteria.

This service needs to fetch all the **criteria** (division, market segment, funding type, effective date, renewal type) for UI to consume.

**Description:**

Initiate Renewal – Load Initiate Criteria

**Assumptions:**

Criteria provided will have all possible criteria results. This story will not talk about any business rule-based criteria to be pulled but is generic.

A Micro Service will be defined underlying to this service to retrieve the data from the database.

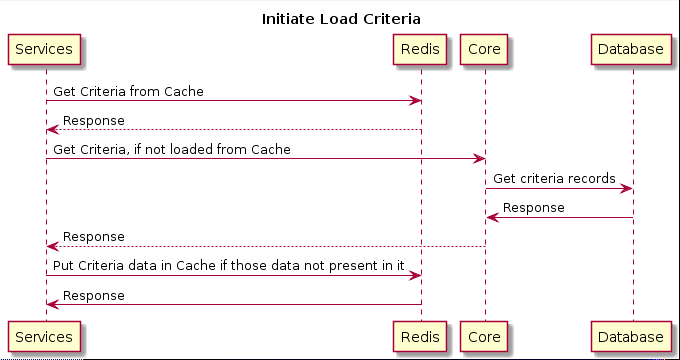
Audit service is part of another user story.

The system-specific exceptions that occur will be captured as part of another user story.

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **User story title** | **User story description** | **Notes** |
| **1.** | US#14310 | Initiate Renewal – Load Initiate Criteria | 1. Load criteria from Cache 2. If not found in cache, then request Core micro services to load criteria from database. 3. Put the data in cache which is not present in it. 4. Spilt the data as per fields. 5. Send the response to UI |

**Sequence diagram for Load Initiate Criteria:**

The below diagram explains the interaction between the Services module with Redis, Core and DB. This diagram briefs about how the criteria is fetched from Cache in case of availability and unavailability. The interactions are explained briefly in the below steps.



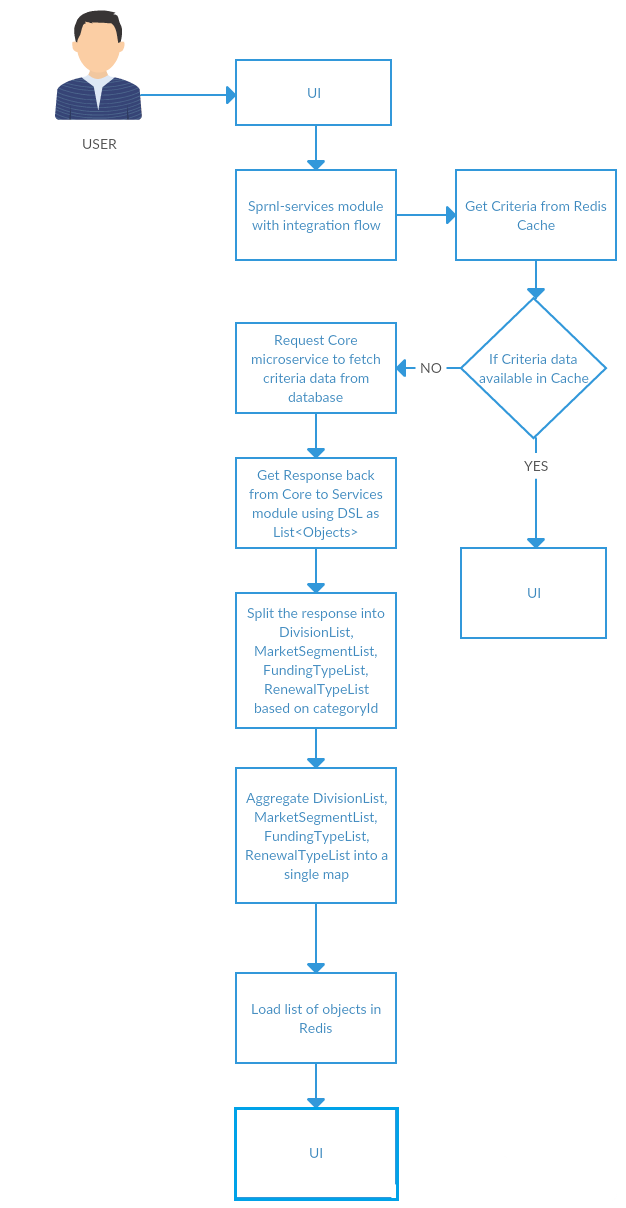
**Step1:** Request is sent to Redis cache for Criteria.

**Step 2:** Redis returns the criteria in case of availability.

**Step 3:** In case of unavailability of criteria in cache request is sent to core and core connects with DB and gets the data

**Step 4:** cache is updated with the data and the same is loaded into the UI dropdowns.

**Flowchart for Load Initiate Criteria:** The below flowchart explains the interaction between the Services module with Redis, Core and DB for loading the criteria in UI dropdown fields.



**US #QRE-14314 GR: Renewal service - Initiate renewals**

**Summary:**

This Service will check the User role and initiate the renewals if role permits to initiate (Both Batch renewal/individual renewal with all input criteria. Once validated this service needs to call the create batch id service to acquire batched and invoke produce message to queue. AS part of response to ui

If successful, send batch id, if failure occurs send error response.

**Description:**

This service will check the user role and initiate the renewals, if role permits to initiate (both batch renewal / individual renewal) with all input criteria. Once validated, this service needs to call the create batch id service to acquire batch id and invoke produce message to queue. As part of response to UI

if success, send batch id.

If failure occurs. Send error response.

**Assumptions:**

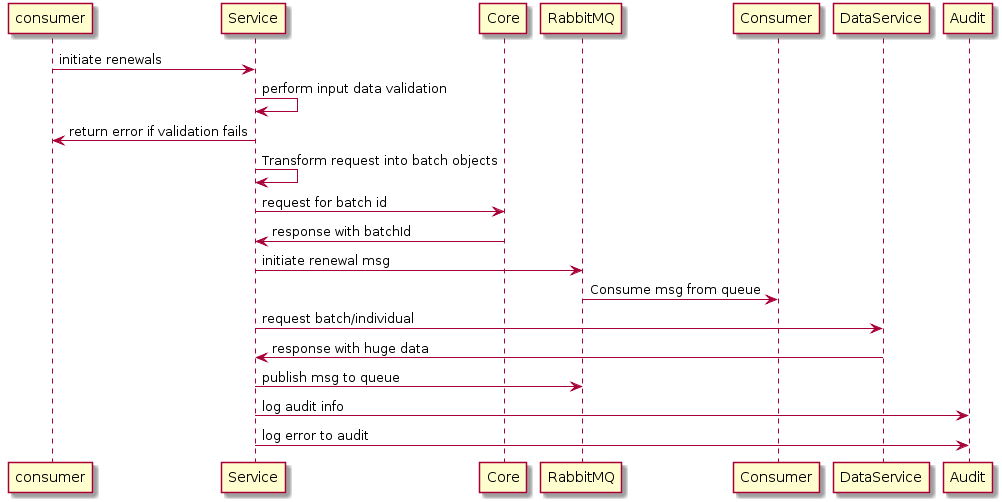
The micro service needed to create batched is already defined. Also, the service to load criteria is also defined.

Any needed business rule validations are handled in a separate user story e.g. check for rating factors. Using test ids until actual user ids are defined.

|  |  |  |  |
| --- | --- | --- | --- |
| **S. No** | **User story title** | **User story description** | **Notes** |
| 1 | QRE – 14314 - GR: Renewal service - Initiate renewals | This service will check the user role and initiate the renewals, if role permits to initiate (both batch renewal / individual renewal) with all input criteria. Once validated, this service needs to call the create batch id service to acquire batch id and invoke produce message to queue. As part of response to UI  if success, send batch id.  If failure occurs. Send error response. | To initiate the renewals process by underwriter or actuary user for batch/ individual based on the role. |

**Sequence diagram for Initiate renewals:**

The below diagram explains the interaction between the User with Services, Core, Rabbit MQ and Audit. This diagram briefs about how the Renewals is initiated and the interactions involved in the same. The interaction steps are explained below.



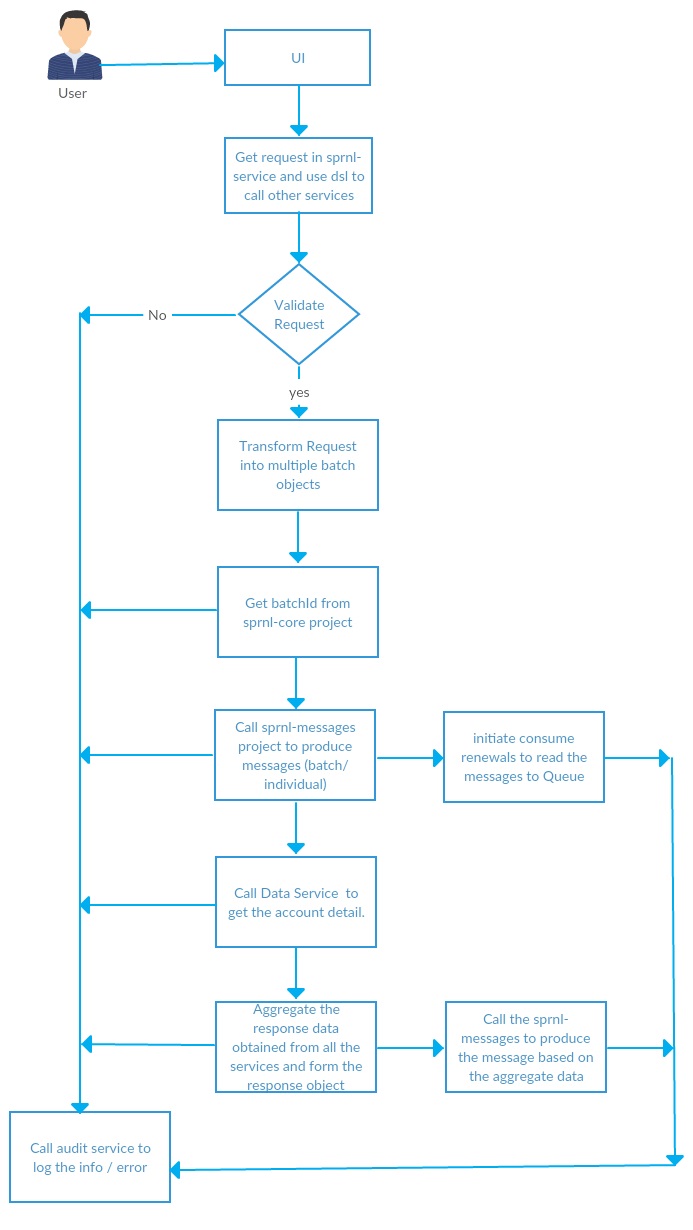
**Step 1:** User sends request to initiate renewals from UI with the Criteria (form fields data). Validation are done on the same and error response is sent back in case of validation failure.

**Step 2:** On Validation success the request is transformed into batch objects and request for batch id is sent to core. Response with Batch id is received.

**Step 3:** Initiate Renewals message is sent to Rabbit MQ from Service and the message is consumed. Request for Batch or Individual queue is sent to Data Service and a huge data is sent as response.

**Step 4:** The received message is published to the queue. Success or error are captured in Audit module.

**Flowchart for Initiate renewals:** The flow chart briefly describes about the initiate renewals process.



# Known issues/limitations

* + 1. Angular 5 is newly released and there might be risks of effort deviation with regards to any stability, Data flow from UI and support issue.
    2. MRDB, Blue Start-Data Lake, BCBS are out of our boundary and there might be risks of effort deviation with regards to any stability and Data flow.

1. Reference Documents
2. Architecture Document: