

Solution Architecture Document

Singapore CMS Eclipse Integration

Version 0.6

Empowered Expertise

**Document Control**

|  |  |
| --- | --- |
| Document Title: | Solution Architecture Document – Singapore CMS Eclipse - Integration |
| Document Owner: | Sacintha Raj – Solution Architect |

**Revision History**

| Version | Date | Status | Description |
| --- | --- | --- | --- |
| 0.1 | 20/09/2022 | Draft | Initial draft |
| 0.2 | 08/10/2022 | Review comments | Review comments provided by Nudi and Chandana |
| 0.3 | 19/10/2022 | Recovery mechanism updated | Recovery mechanism discussed with EY and finalized |
| 0.4 | 26/10/2022 | Review comments | Review comments provided by Nudi |

**Document Distribution**

| Version | Date | Name | Role |
| --- | --- | --- | --- |
| 010.1 | 20/09/2022 | Nudi Sudarshan i | Development Lead |
| 010.1 | 20/09/2022 | Chandana Naveen | Developer |

**Document Approval**

| Version | Date | Name | Role |
| --- | --- | --- | --- |
| 0.10 |  |  |  |
|  |  |  |  |

**Reference Documents**

|  |  |
| --- | --- |
| Document Name | Document Location / Link |
| PSR Mule HLD | [Pending] |
|  | [Pending] |
| CEI WSDL |  |
| Eclipse Claims WSDL |  |
| Eclipse Organisation WSDL |  |

# Table of Contents

[Table of Contents 4](#_Toc125592492)

[1.1 Introduction 5](#_Toc125592493)

[1.2 Document Purpose 5](#_Toc125592494)

[1.3 Business Drivers: 5](#_Toc125592495)

[1.4 Scope 6](#_Toc125592496)

[1.5 Architectural Goals and Constraints 10](#_Toc125592497)

[1.6 Architecture Definition 12](#_Toc125592974)

[1.7 Design 17](#_Toc125592975)

[1.8 Approach 34](#_Toc125592977)

[1.9 Assumptions 34](#_Toc125592978)

[1.10 Dependencies 35](#_Toc125592979)

[2 Appendix 35](#_Toc125592980)

[2.1 Application List 35](#_Toc125592981)

[2.2 Glossary 35](#_Toc125592982)

## Introduction

As part of **CMS Release 2 - Singapore Implementation Project**, TMK Singapore wants to use the CMS system as a one stop solution for it is claims management. As part of this drive, new systems must be added, and some systems must be upgraded.

It is also identified the need for data reconciliation.

## Document Purpose

The purpose of this document is to identify the systems involved in achieving the integration between CMS and Eclipse, and the interactions between them.

This integration will not be used for Tokio Marine Kiln’s London Syndicate business but is required for Singapore Claims, so that Eclipse can feed data to downstream systems e.g., Workday, the General Ledger, Eclipse Reporting.

This document will discuss the systems/application from the NTT suite, which is part of the whole implementation, that will help achieve the intended goal

This document forms the basis for upcoming design documents and implementation

## Business Drivers:

TMK Singapore business users intend to use CMS as their only system to create and manage claims. To cater to their needs, CMS Release 2 - Singapore Implementation Project is undertaken.

This will enable Singapore users to have the below capabilities

* Singapore claims handlers will utilize the enhanced functionality within CMS to support their claims management process.
* CMS will send details regarding Singapore claims to Eclipse upon claim creation, claim update, claim closure, payment creation and financial update events.
* CMS will also utilize the Organization Node to ensure payee and contact information is sync’d across the two systems
* Claims handlers will input financials into CMS and these will be fed into Eclipse to aid any onward processing and decision making
* Claims handlers will have the ability to instruct payments from CMS
* Finance will continue to utilize Eclipse to process payments
* Reconciliation of data between CMS and Eclipse

## Scope

The scope of the document identifies and elucidate the technical solution provided by NTT as part of the whole implementation

Graphical user interface, application

Description automatically generated

This document explains the technical solution and the architecture at a high level. There are three systems identified in NTT landscape that either needs a uplift of new implementation

1. CEI - CMS will communicate to Eclipse via CEI. There are seven APIs to cater to. A new system called CEI – CMS Eclipse Integrator has to be built. This is new addition to the NTT Estate
2. PSR - Existing PSR API to incorporate additional data in both request and response, must be upgraded. Currently it serves TMK London. It needs to be upgraded to cater to Singapore TMK as well.
3. DMS - Existing SharePoint mule Application which serves London TMK as a document management system. This needs to be upgraded to cater to Singapore.

### CMS Eclipse Integrator

### Business scope

CMS Eclipse Integrated (CEI) which integrates CMS and Eclipse. The following APIS are now supported by CEi

1. CreateClaim - This flow will be triggered when a claim has been created within CMS.
2. UpdateClaim - This flow will be triggered when a significant change, has been made to a claim OR when a claim’s status has been changed within CMS.
3. CreateMovement - This flow will be triggered when a financial movement is created within CMS.
4. PostClaimMovementToLedger - This flow will be triggered when a claim movement needs to be posted to the Ledger e.g., when a payment or recovery has been approved or when a payment has been voided or stopped within CMS, post the financial escalation process
5. UpdateClaimMovementBrokerTransRef - This flow will be triggered when a new payment has been added to a claim within CMS.
6. CreateOrganisationNode - This flow will be triggered when a new Organisational contact has been added to a claim within CMS.
7. UpdateOrganisationNode - This flow will be triggered when an Organisational contact has been updated on a claim within CMS.

Additionally, another API is used to weed out duplicate claim and movements

* Lookup Api- This flow is triggered to verify any previous organizations have been created.
* GetClaim: - This flow is triggered to verify any previous Claims have been created or movements have been triggered. If we need to retrieve

**Error Handling and recovery**

Below Technical errors while invoking CMS or Eclipse are handled at CEI Layer

1. Time Out in CMS or Eclipse
2. Temporarily outage of Eclipse or CMS
3. Permanent outage of Eclipse or CMS
4. Duplicate Request handled at CEI

Any other errors that can be progressed will be passed back to CMS.

EX: If a movement is already posted to ledger, the post movement to Ledger Request would fail with an exception. These Kind of exceptions would be passed on to CMS

**Data Reconciliation**

Data from request and response would be captured which would be used to support reconciliation.

### Technical Scope

This is entirely a new implementation

### Business Scope

### Technical Scope

Existing systems must be upgraded to cater to Singapore and are listed below

1. Existing ETL Jobs
2. Mule PSR Application

### DMS - SharePoint Mule

### Business Scope

Existing DMS system should be able to cater to Singapore claims.

### Technical Scope

Existing systems must be upgraded to cater to Singapore and are listed below

1. Existing SharePoint system
2. Existing Mule SharePoint Application

## 

### CMS Eclipse Integrator

The below technical goals are taken into consideration

1. A single message from CMS which is accepted for processing should not be lost
2. Prevent any duplicate messages from entering and being processed in the system
3. All recoverable technical issues with Eclipse or CMS should be effectively managed by Fuse
4. Communication to support team should be facilitated when there is failure in any system beyond accepted SLA

To achieve the above goals, the below software is used

1. Jboss Fuse should be configured with HA configuration
2. Amazon MQ (Active MQ) should be configured for HA
3. Dynamo Db for storing Audit data, Reconciliation data and details of request and response
4. AWS S3 for storing request and response.

### PSR

1. Existing Policy Search and Retrieve Mule Application should be modified so that it caters both to London and Singapore
2. Existing ETL jobs that populate KDR should be updated for Singapore data.

### DMS

1. Existing SharePoint application should be modified so that it caters both to London and Singapore
2. Existing Document Management System (Sharepoint) should be enhanced to provide Document Management to both London and Singapore

### Constraints

### Error handling

1. Any technical errors that are recoverable by the deployment architecture would be handled by CEI
2. supported TMK

as it is accessed by LDAP Security

As in existing applications c

The default configuration would be suitable for the volume of Requests.

### Scalability

The default configuration would suffice for the volume of requests











## Architecture Definition

### CMS Eclipse Integrator

Diagram

Description automatically generated

The Component breakdown is as follows

1. CMS -Guide wire Claim system
2. CEI -CMS Eclipse Integrator
   1. Middleware cluster Jboss Fuse .- This is an integration software which handles data transformartion and to handle endpoint retreis and connection recovery.
   2. Active MQ – Messaging Queues for effective data processing
   3. AWS S3 – S3 bucket to store request and response.
   4. DynamoDB
      1. Table to capture data to support reconcilation at a later date
      2. Table to capture Audit information
      3. Table to store Request and Response for later use

The Component Interaction as below

1. Fuse interacts with CMS and Eclipse and to receive and process request and responses
2. Fuse utilizes Active MQ to store and forward messages at different stages of the request response lifecycle.
3. Fuse stores data for various purposes in DynamoDB
4. Fuse stores raw and processed request and response in AWS S3 Bucket.

### PSR Mule

Graphical user interface, application

Description automatically generated

This is an existing system that is currently catering for London.

As depicted in the above pic the ETL process and Mule PSR would be upgraded to cater for Singapore

1. ETL Process
   1. Additional data from Eclipse would be pulled from Eclipse database and pushed into KDR summary table by an overnight job.
2. Mule PSR is upgraded to have the below enhanced functionalities
   1. Singapore policies are identified using either Legal Branch or Dummy Syndicates in case Legal branch is null.
   2. UMR for Singapore policies are expected to be always null. London implementation has some validations on UMR not being null. These implementations are modified so that such validations are skipped for Singapore policies.
   3. The Branch and Dummy Syndicate values are parametrized, and the implementation is future proofed, so that known branch structure changes (as attached) when implemented in production, would not call for PSR Mule changes.



### SharePoint Mule

Graphical user interface, application

Description automatically generated

This is an existing system that is currently catering for London.

As depicted in the above pic the SharePoint DMS and SharePoint Mule application would be upgraded to cater for Singapore

1. DMS Upgrade

DMS stores documents based on configured templates which can be retrieved later

DMS stores documents based on configured templates which can be re

1. Sharepoint Mule Application

This Mule application to be updated to manage document upload, retrieve, and delete on Singapore SharePoint DMS for Singapore claim related documents.

## Design

In this section we will discuss the implementation of all the components described in Architecture Definition section.

We will only discuss the implementation of CMS Eclipse Integrator as the other systems need

enhancements alone

### Design Objectives

* CEI should easily be extendable to other divisions and contract types in future
* CEI not to hold any major business/functional rule implementation
* CEI would be sitting as an EIB facilitating interface between CMS and Eclipse
* CEI solution is scalable to expected requests volume

### Design Overview

A picture containing diagram

Description automatically generated

The whole flow of request and response between CMS and Eclipse via CEI is dissected into three major flows.

1. CMS Request to CEI

CMS Invokes CEI -Fuse via HTTPS/SOAP call. At CEI, the request is validated and backed up in a backup table and posted to a request processing queue for further processing

1. CEI Request to Eclipse

CEI picks up the request from the request processing Queue (AMQ) and transformed to another request That will be accepted by Eclipse and posted on to Eclipse Request Queue. From Eclipse Request Queue (AMQ) the request is used to Invoke a HTTP SOAP service on Eclipse. The response from eclipse is validated and stored in a backup table and posted on to a response processing queue (AMQ).

1. CEI Response to CMS

CEI picks the eclipse response from Response processing queue and transforms to a CMS response and this CMS response is posted on to a CMS Queue (AMQ). Fuse picks the CMS Response from CMS Queue and send to CMS by invoking a callback HTTP SOAP Service exposed by CMS

Each stage the request flows we use Active MQ, to save the state of the request/response

We use Data base for Audit, Reconciliation and Data Backup for case of crash and recovery

Below Technical errors while invoking CMS or Eclipse are handled at CEI Layer

1. Time Out in CMS or Eclipse
2. Temporarily outage of Eclipse or CMS
3. Permanent outage of Eclipse or CMS

### Detailed Design

Detail Design is divided into flowcharts and flows.

1. Flowcharts show the flows of request and response between CMS and Eclipse via CEI
2. Flows detail different scenarios

**Flowcharts**

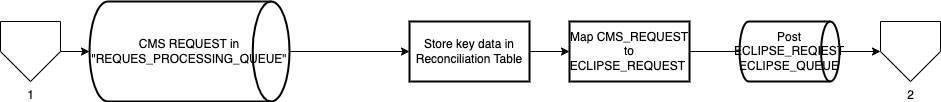
1. CMS to FUSE Request

**Diagram

Description automatically generated**

1. Map and post to Eclipse Queue

Map the request to eclipse request and post to eclipse queue



1. Invoke Eclipse and handle different failure scenarios
   1. Successful invocation resulting in flow number 9. Received a successful Response and passed to Flow 9
   2. Invalid Response Resulting in flow number 3. Email sent to Eclipse and Request posted to backup queue
   3. Response time out resulting in flow number 3. Email sent to Eclipse and Request posted to backup queue
   4. Eclipse unavailability resulting flow 3. Email sent to Eclipse and Request posted to backup queue
   5. Response time out resulting in flow 5 since the same request cannot be posted again. Flow 5 handles duplicate request
   6. Inherently handles the temporary eclipse unavailability

**Diagram

Description automatically generated**

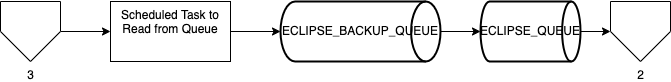
1. Store Eclipse response and pass it on”
   1. If there is any failure in s3 storage or Data base storage

**Diagram

Description automatically generated**

1. Reload Request from Backup queue to eclipse queue to continue processing

Reload Request from Backup queue to eclipse queue to continue processing

****

1. Eclipse Response mapped and posted to CMS Queue

**Diagram

Description automatically generated**

1. Eclipse Duplicate Request strategy
   1. Alternate request is used and confirm the original request has been successful and results in flow 7 where the original path of response flow continues
   2. Alternate request is used and confirm the original request has not been successful and results in flow 2. Post the ECLIPSE\_REQUEST to Eclipse queue and for processing the request again
   3. The alternate request also failed and results in flow 6. The original request is stored in a DB for Future processing

Diagram

Description automatically generated

1. Request from table to Duplicate Flow

**Diagram

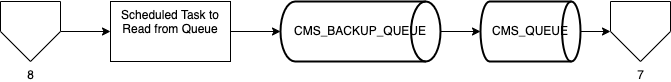
Description automatically generated**

1. Invoke CMS
   1. Successful invocation of CMS resulting in completion of flow
   2. Any recoverable error results in flow 8 which backs the CMS Response in a backup queue

**A picture containing text, clock

Description automatically generated**

1. CMS Response from backup queue
   1. Request moved from backup Queue to CMS Queue

****

The Flow are described below.

**Happy Path Request Flow**

The Flow of request is as below

1. CMS invokes the SOAP webservice exposed by CEI(Fuse).
2. The request is validated and stored in a S3 bucket, and the path is stored in a Dynamo DB table “DATA\_STORE” for later consumption.
3. The request is posted on to a queue “REQUEST\_PROCESSING\_QUEUE”
4. An acknowledgment is sent back to CMS.
5. The Request is picked from the “REQUEST\_PROCESSING\_QUEUE”.
6. The request is mapped to an “Eclipse Request”.
7. This “Eclipse Request” is posted on to a “ECLIPSE\_QUEUE”
8. The “Eclipse Request” is then used to invoke an API on the Eclipse.
9. The “Eclipse Response” is received from Eclipse.
10. The “Eclipse Response” is validated and stored in a S3 bucket, and the path is updated in Dynamo DB table “DATA\_STORE” for later consumption.
11. The “Eclipse Response” is posted to “RESPONSE\_PROCESSING\_QUEUE”.

**Happy Path Response Flow**

The Flow of response is as below

1. The “Eclipse Response” is picked from the “RESPONSE\_PROCESSING\_QUEUE”.
2. The Key fields from response are send to a DynamoDB Table “Reconciliation Table”
3. The Eclipse Response is mapped to a “CMS Response”
4. The “CMS Response” is posted on to a “CMS\_QUEUE”
5. The “CMS Response” is picked from the “CMS\_QUEUE” and used to invoke the CMS Callback APIs
6. Upon successful acknowledgment Response from CMS the request and Response is cleared from the “DATA\_STORE” and marked complete on the “DATA\_STORE”

**Alternative Flows**

The Above two flows are successful Request and Response flows.

Below this section we will discuss the alternative flows arising out of error scenarios

**CMS Invalid Request Flow**

1. CMS invokes the SOAP webservice exposed by CWA(Fuse).
2. The request is validated and found to be an error.
3. A soap fault is sent back to CMS

**CMS Request storage to S3 bucket has failed**

1. CMS invokes the SOAP webservice exposed by CWA(Fuse).
2. The request is validated.
3. The request is stored in a s3 bucket
4. A soap fault is sent back to CMS

**CMS duplicate Request Flow**

1. CMS invokes the SOAP webservice exposed by CWA(Fuse).
2. The request is validated
3. The request is stored in a S3 Bucket
4. The path reference is stored to the “DATA\_STORE”
5. A primary key validation occurs which denotes a duplicate request.
6. A soap fault is sent back to CMS

**Eclipse Temporarily Unavailable Request Flow**

1. CMS invokes the SOAP webservice exposed by CWA(Fuse).
2. The request is validated and stored in a S3 bucket, and the path is stored in a Dynamo DB table “DATA\_STORE” for later consumption.
3. The request is posted on to a queue “REQUEST\_PROCESSING\_QUEUE”
4. An acknowledgment is sent back to CMS.
5. The Request is picked from the “REQUEST\_PROCESSING\_QUEUE”.
6. The Key fields from request are send to a DynamoDB Table “Reconciliation Table”
7. The request is mapped to an “Eclipse Request”.
8. This “Eclipse Request” is posted on to a “ECLIPSE\_QUEUE”
9. The “Eclipse Request” is then used to invoke an API on the Eclipse.
10. If the Eclipse is unavailable when invoking the API.
11. This “Eclipse Request” is tried after some interval of time
12. Likewise step 10 and 11 are repeated for three times.
13. Before the last attempt is used Eclipse is available and it is invoked successfully
14. The “Eclipse Response” is received from Eclipse.
15. The “Eclipse Response” is validated and stored in a S3 bucket, and the path is updated in Dynamo DB table “DATA\_STORE” for later consumption.
16. The “Eclipse Response” is posted to “RESPONSE\_PROCESSING\_QUEUE”.

**Eclipse Permanently Unavailable Request Flow**

1. CMS invokes the SOAP webservice exposed by CWA(Fuse).
2. The request is validated and stored in a S3 bucket, and the path is stored in a Dynamo DB table “DATA\_STORE” for later consumption.
3. The request is posted on to a queue “REQUEST\_PROCESSING\_QUEUE”
4. An acknowledgment is sent back to CMS.
5. The Request is picked from the “REQUEST\_PROCESSING\_QUEUE”.
6. The Key fields from request are send to a DynamoDB Table “Reconciliation Table”
7. The request is mapped to an “Eclipse Request”.
8. This “Eclipse Request” is posted on to a “ECLIPSE\_QUEUE”
9. The “Eclipse Request” is then used to invoke an API on the Eclipse.
10. If the Eclipse is unavailable when invoking the API.
11. This “Eclipse Request” is tried after some interval of time
12. Likewise step 10 and 11 are repeated for three times.
13. All attempts are exhausted and still the eclipse is down. And the message is sent to “ECLIPSE\_BACKUP\_QUEUE”
14. Now an Email is sent to respective personal so that Eclipse can be bought back to operation.
15. We now wait for an email notification from Eclipse to state that Eclipse is up and running.
16. The request is moved from “ECLIPSE\_BACKUP\_QUEUE” to “ECLIPSE\_QUEUE”. This is done by a scheduler set to run at separate intervals of time
17. The “Eclipse Request” is then used to invoke an API on the Eclipse.
18. The “Eclipse Response” is received from Eclipse.
19. The “Eclipse Response” is validated and stored in a S3 bucket, and the path is updated in Dynamo DB table “DATA\_STORE” for later consumption.
20. The “Eclipse Response” is posted to “RESPONSE\_PROCESSING\_QUEUE”.

**Eclipse Times Out Request Flow**

1. CMS invokes the SOAP webservice exposed by CWA(Fuse).
2. The request is validated and stored in a S3 bucket, and the path is stored in a Dynamo DB table “DATA\_STORE” for later consumption.
3. The request is posted on to a queue “REQUEST\_PROCESSING\_QUEUE”
4. An acknowledgment is sent back to CMS.
5. The Request is picked from the “REQUEST\_PROCESSING\_QUEUE”.
6. The Key fields from request are send to a DynamoDB Table “Reconciliation Table”
7. The request is mapped to an “Eclipse Request”.
8. This “Eclipse Request” is posted on to a “ECLIPSE\_QUEUE”
9. The “Eclipse Request” is then used to invoke an API on the Eclipse.
10. Eclipse times out before sending a response back to Fuse.
11. Fuse has to confirm if the request is successful at Eclipse before resending the request.
12. For each API we need a different strategy to test if the previous request was successful.
13. These Strategies are discussed in the section “Duplicate Resolving Strategy”
14. Based on the outcome of the strategy, if the previous request is a success the below is the onward flow
    1. The “Eclipse Request” is removed from the “ECLIPSE\_QUEUE”
15. Based on the outcome of the strategy, if the previous request was a failure the steps following this step is executed flow
16. The “Eclipse Request” is then used to invoke an API on the Eclipse.
17. The “Eclipse Response” is received from Eclipse.
18. The “Eclipse Response” is validated and stored in a S3 bucket, and the path is updated in Dynamo DB table “DATA\_STORE” for later consumption.
19. The “Eclipse Response” is posted to “RESPONSE\_PROCESSING\_QUEUE”.

**Invalid Eclipse Response Flow**

1. CMS invokes the SOAP webservice exposed by CWA(Fuse).
2. 2. The request is validated and stored in a S3 bucket, and the path is stored in a Dynamo DB table “DATA\_STORE” for later consumption.
3. The request is posted on to a queue “REQUEST\_PROCESSING\_QUEUE”
4. An acknowledgment is sent back to CMS.
5. The Request is picked from the “REQUEST\_PROCESSING\_QUEUE”.
6. The Key fields from request are send to a DynamoDB Table “Reconciliation Table”
7. The request is mapped to an “Eclipse Request”.
8. This “Eclipse Request” is posted on to a “ECLIPSE\_QUEUE”
9. The “Eclipse Request” is then used to invoke an API on the Eclipse.
10. If the Eclipse is unavailable when invoking the API.
11. This “Eclipse Request” is tried after some interval of time
12. Likewise step 10 and 11 are repeated for three times.
13. Before the last attempt is used Eclipse is available and it is invoked successfully
14. The “Eclipse Response” is received from Eclipse.
15. The “Eclipse Response” is Validated and found to be invalid
16. An email is sent to the relevant stakeholders and manually resolve the situation
17. When this issue is resolved, Eclipse Times Out Request Flow is executed from step 11.

**S3 Bucket or DynamoDB unavailable for response Storage**

The Flow of request is as below

1. CMS invokes the SOAP webservice exposed by CWA(Fuse).
2. The request is validated and stored in a S3 bucket, and the path is stored in a Dynamo DB table “DATA\_STORE” for later consumption.
3. The request is posted on to a queue “REQUEST\_PROCESSING\_QUEUE”
4. An acknowledgment is sent back to CMS.
5. The Request is picked from the “REQUEST\_PROCESSING\_QUEUE”.
6. The request is mapped to an “Eclipse Request”.
7. This “Eclipse Request” is posted on to a “ECLIPSE\_QUEUE”
8. The “Eclipse Request” is then used to invoke an API on the Eclipse.
9. The “Eclipse Response” is received from Eclipse.
10. The “Eclipse Response” is validated and stored in a S3 bucket, and the path is updated in Dynamo DB table “DATA\_STORE” for later consumption.
11. If S3 bucket or Dynamo Db is unavailable during response storage the response is posted on to the “RESPONSE\_BACKUP\_QUEUE”
12. An Email is sent to Fuse support
13. A scheduler is run a specific times of the day and will pick the Eclipse response from “RESPONSE\_BACKUP\_QUEUE” and steps from 10 is repeated again.
14. The “Eclipse Response” is posted to “RESPONSE\_PROCESSING\_QUEUE”.

**CMS Temporarily Unavailable Response Flow**

1. The “Eclipse Response” is picked from the “RESPONSE\_PROCESSING\_QUEUE”.
2. The Key fields from response are send to a DynamoDB Table “Reconciliation Table”
3. The Eclipse Response is mapped to a “CMS Response”
4. The “CMS Response” is posted on to a “CMS\_QUEUE”
5. The “CMS Response” is picked from the “CMS\_QUEUE” and used to invoke the CMS Callback APIs
6. The “CMS Response” is then used to invoke an API on the CMS.
7. If the CMS is unavailable when invoking the API.
8. This “CMS Response” is tried after some interval of time
9. Likewise step 7 and 8 are repeated for three times.
10. Before the last attempt is used CMS is available and it is invoked successfully
11. Upon successful acknowledgment Response from CMS the request and Response is cleared from the “DATA\_STORE” and marked complete on the “DATA\_STORE”

**CMS Permanently Unavailable Response Flow**

1. The “Eclipse Response” is picked from the “RESPONSE\_PROCESSING\_QUEUE”.
2. The Key fields from response are send to a DynamoDB Table “Reconciliation Table”
3. The Eclipse Response is mapped to a “CMS Response”
4. The “CMS Response” is posted on to a “CMS\_QUEUE”
5. The “CMS Response” is picked from the “CMS\_QUEUE” and used to invoke the CMS Callback APIs
6. The “CMS Response” is then used to invoke an API on the CMS.
7. If the CMS is unavailable when invoking the API.
8. This “CMS Response” is tried after some interval of time
9. Likewise step 7 and 8 are repeated for three times.
10. All attempts are exhausted and still the CMS is down. and the Message is sent to “CMS\_BACKUP\_QUEUE”
11. Now an Email is sent to respective personal so that CMS can be bought back to operation.
12. We now wait for an email notification from CMS to state that CMS is up and running.
13. The request is moved from “CMS\_BACKUP\_QUEUE” to “CMS\_QUEUE”. This is done by a scheduler set to run at separate intervals of time
14. The “CMS Response” is then used to invoke an API on the Eclipse.
15. Upon successful acknowledgment Response from CMS the request and Response is cleared from the “DATA\_STORE” and marked complete on the “DATA\_STORE”

**Fuse Failures**

There is a possibility that Fuse can be down.

When the error is fixed and Fuse is bought back to operation, Fuse need to know where it needs to start.

Here is where an Audit table comes to play.

The Audit Table is updated as the Request and Response travels through the fuse.

|  |  |
| --- | --- |
| REQUEST\_RESPONSE\_STAGE | Value to be Stored in Audit Table |
| Request Validated and stored in the “DATA\_STORE” table and posted in REQUEST\_PROCESSING\_QUEUE | REQUEST\_VALIDATED\_STORED |
| Acknowledgment successfully sends to CMS | CMS\_ACKNOWLEDGEMENT\_SENT |
| Request Mapped and Posted in ECLIPSE\_QUEUE | REQUEST\_ECLIPSE\_QUEUE |
| Request Posted to Eclipse | INVOKED\_ECLIPSE |
| Response Received from Eclipse | ECLIPSE\_RESPONSE\_RECEIVED |
| Response Mapped and posted to CMS\_QUEUE | RESPONSE\_CMS\_QUEUE |
| Response posted to CMS | INVOKED\_CMS |
| Acknowledgement Received from CMS | CMS\_ACKNOWLEDGEMENT\_RECEIVED |
| “ECLIPSE REQUEST” posted to “ECLIPSE\_BACKUP\_QUEUE” | REQUEST\_POSTED\_TO\_ECLIPSE\_BACKUP\_QUEUE |
| “CMS Response” posted to “CMS\_BACKUP\_QUEUE” | RESPONSE\_POSTED\_TO\_CMS\_BACKUP\_QUEUE |
| Exception received from Eclipse | EXCEPTION\_RECEIVED\_FROM\_ECLIPSE |
| Exception received from CMS | EXCEPTION\_RECEIVED\_FROM\_CMS |
| Retrieved from ECLIPSE\_BACKUP queue | RETRIEVED\_FROM\_ECLIPSE\_BACKUP\_QUEUE |
| Retrieved from CMS\_BACKUP queue | RETRIEVED\_FROM\_CMS\_BACKUP\_QUEUE |
| Eclipse Response posted on to “RESPONSE\_BACKUP\_QUEUE” | POSTED\_TO\_RESPONSE\_BACKUP\_QUEUE |

When Fuse is restarted after a crash the stages that would be of interest are as below

1. INVOKED\_ECLIPSE
   1. When this the last entry in the Audit Table, the flow that will be used upon resuming is “Eclipse Times Out Request Flow”. The entries in Audit Table will be marked as second run
2. INVOKED\_CMS
   1. When this is the last entry in the Audit Table, the flow that will be used upon resuming is “Standard response flow”. The entries in Audit Table will be marked as second run

When Fuse is restarted a scheduler will run once to find all data has the above two Audit stages.

In this scenario Scheduler will be a starting point of the requests flow instead of CMS.

**Strategy to Decide Resubmitting Request to Eclipse**

There are 7 APIs in question and each API will have a different Strategy.

1. CreateClaim – In order to resubmit this request, we will use a GetClaim call to figure if this has been already submitted. This GetClaim call will give the details of the Claim created if it has been already created
2. UpdateClaim – This can be resubmitted without any issue
3. CreateMovement - In order to resubmit this request, we will use a GetClaim call to figure if this has been already submitted. This GetClaim call will give the details of the Movement created if it has been already created. In get claim response there will be a list of movements.The movement that has the created date set after request time is the movement that has been created

.

1. PostClaimMovementToLedger – If this is resubmitted, two things will happen
   1. If the previous request is already successful, we will get a business error stating that it is already processed. In this scenario the response that will be send to CMS will inform then it has been processed successfully
   2. If the previous request was not successful than we will get a positive response from eclipse that can be passed on to CMS
2. UpdateClaimMovementBrokerTransRef - If this is resubmitted, two things will happen
   1. If the previous request is already successful, we will get a business error stating that it is already processed. In this scenario the response that will be send to CMS will inform then it has been processed successfully
   2. If the previous request was not successful than we will get a positive response from eclipse that can be passed on to CMS
3. CreateOrganisationNode - In order to resubmit this request, we will use a look up call to figure if this has been already submitted. This lookup call will give the details of the Organization created if it has been already created

.

1. UpdateOrganisationNode - This can be resubmitted without any issue.

**Table Structures**

**Audit Table**

1. Message\_Id –The message Id of the Message. This is first received from CMS
2. Operation Name: Name of the operation
3. Current Stage-The stage that the processing of the message is at
4. Run No – this denotes how what is number of a times the above request or response flow has been run.
5. Retry No: this exception retries to contact CMS or Eclipse. Other times it will be empty
6. CreatedTime: The time of this audit
7. Details: Details like Exception will be stored in here.

**DATA\_STORE**

1. Message\_Id: The message Id of the Message. This is first received from CMS
2. Raw\_Request: Validated Request from CMS
3. Raw\_Response:Validated Response from Eclipse
4. Mapped\_Eclipse\_Request: Mapped Eclipse Request
5. Mapped\_CMS\_RESPONSE: Mapped CMS Response
6. Completed : Boolean if the entire process is completed
7. Request\_Time : Time of request
8. Reponse\_time : Time of response

**RECONCILIATION\_TABLE**: The Dynamo DB table for data reconciliation support would have the below columns

1. MessageId – this would the id of every message. This Message id will be the request key data from CMS
2. OperationName – Operation Name of this request and response name
3. RequestType – It is the Request type. It has same value as OperationName. The same request can be used for two different operations hence we have it in separate fields.
4. ResponseFieldName – The name of the field that will identify the entity in Eclipse
5. ResponseFieldValue -The value of the field that will identify the entity in Eclipse

|  |  |  |
| --- | --- | --- |
| Operation Name | Request Type | Response Field Name |
|  |  |  |
| CreateClaim | CreateClaim | ClaimId |
| UpdateClaim | UpdateClaim | ClaimId |
| CreateMovement | CreateMovement | MovementId |
| PostClaimMovementToLedger | PostClaimMovementToLedger | MovementId |
| UpdateClaimMovementBrokerTransRef | UpdateClaimMovementBrokerTransRef | BrokerRef |
| CreateOrganisationNode | CreateOrganisationNode | OrganisationId |
| UpdateOrganisationNode | UpdateOrganisationNode | OrganisationId |
| GetClaim | CreateMovement | Movemnet Id |
| GetClaim | CreateClaim | ClaimId |
| Lookup | CreateOrganisationNode | OrganisationId |

## Approach

### Technology

| **Technology Stack** | **Components used in** |
| --- | --- |
| RHEL 7.9 | CEI, PSR, Sharepoint Mule |
| JBoss EAP 7.4 | CEI |
| RedHat Fuse 7.10 | CEI |
| Mule 3.9 | PSR, Sharepoint Mule |
| Hashi Vault (used by Git Pipelines) | CEI, PSR, Sharepoint Mule |
| CyberArk | CEI, PSR, Sharepoint Mule |
| GitLab/pipelines | CEI, PSR, Sharepoint Mule |
| MS SQL 2017 CU24 | PSR, Sharepoint Mule |
| Amazon Dynamo DB | CEI |

### Common Services

1. CyberArk\_Integration\_Web Fuse service – Common service to fetch password from CyberArk for given account. NA. CWA do not control any business data.

### System Management, Monitoring, and Administration

1. Each CEI service within the scope will be tested for successful execution in the ST Environment. (Can be misunderstood as a different env) Successful Integration and integrity of data exchanged will be tested. No checks on data will be performed.
2. CEI monitoring and management will be supported by TMK IS

## Assumptions

1. Complete environment would be provisioned from Development phase. Development Integration testing is critical as CWA includes multiple end point interaction.
2. All the network firewall related changes to enable seamless communication of CEI with other end points will be supported by TMK IS
3. All credentials can be stored in CyberArk and corresponding CyberArk accounts would be created
4. Eclipse does not use security to invoke service or access any records
5. The order of requests is determined by CMS and Fuse would treat each request individually without any dependency on any other requests.
6. The SLAs would be finalized for number of retries, interval of retries.
7. Verisk has advised not to validate Response
8. Claim and related records created in eclipse by CMS are assumed to be like the existing records pulled by KDR

## Dependencies

1. Complete integrated end to end environment including Eclipse, amazon load balancer, Amazon DynamoDB , MSSQL Database (as applicable) would be provisioned from Development phase. Development Integration testing is critical as CWA includes multiple end point interaction.
2. All the network firewall related changes to enable seamless communication of Fuse with other end points will be addressed by TMK IS
3. Eclipse Api must be tested before LLD
4. KDR Changes to be provided by NTT ETL Team
5. SharePoint verification and configuration by TMK
6. EY to raise any concerns with the WSDL of CWA, PSR and Mule Sharepoint
7. Get WSDL for Lookup API from Verisk
8. A valid sample request to get organization node using Lookupservice , from verisk
9. A valid sample request to get Claim using Getclaim without using claim id, from verisk
10. Jboss Fuse and Active MQ is configured in HA Mode
11. Any files uploaded to S3 Bucket is encrypted before uploading.
12. Any files read from S3 Bucket should be decrypted.

# Appendix

1. None

## Application List

1. CIA
2. PSR
3. SharePoint Mule
4. KDR Including with ETL process
5. Amazon Dynamodb

## Glossary

|  |  |
| --- | --- |
| Name | Description |
| HLD | High Level Design |
| CMS | Claims Management System |
| CEI | Camel Eclipse Integrator |
| GWLM | Guidewire London Market Extension Pack |
| DMS | Document Management System |
| EY | Ernst & Young. Third party providing CMS |
| VERISK | Third party providing Eclipse |
| CYBERARC | Sensitive information management system |
| KDR | ETL system |
| PSR | Policy search and Retrieve |