Solution Design Document

for Zurich Travel Insurance Management System

Contents

[Use Case Overview 2](#_Toc126153624)

[Technical Design 2](#_Toc126153625)

[Components/ Mule Application 3](#_Toc126153626)

[List of applications that are involved in integration 3](#_Toc126153627)

[Non-Functional Requirements 4](#_Toc126153628)

[Security 4](#_Toc126153629)

[Client Id Enforcement 4](#_Toc126153630)

[Token Verification (OKTA OAuth2.0) 4](#_Toc126153631)

[Sequence Diagram 4](#_Toc126153632)

[Quotation API 5](#_Toc126153633)

[Insurance API 6](#_Toc126153634)

[List of API endpoints involved in the integration. 6](#_Toc126153635)

[API Request and Response 6](#_Toc126153636)

[Exception Management & Error Handling 7](#_Toc126153637)

[System Error 7](#_Toc126153638)

[Validation Error 8](#_Toc126153639)

[Question & Answers 8](#_Toc126153640)

# Use Case Overview

The integration is to facilitate customers to purchase/renew insurance, check their insurance portfolio and submit a claim.

The Integration will be achieved using MuleSoft Anypoint Platform.

# Technical Design

Below is the API let connectivity for travel insurance API Integration

Solution Architecture Diagram

Diagram

Description automatically generated

# Components/ Mule Application

# List of applications that are involved in integration

|  |  |  |
| --- | --- | --- |
| **Mule Applications** | | |
| S.No | Application Name | Description |
| 1 | customer-svc-exp-v1 | This application will be exposed to travel booking application to consume   * Listen to incoming request via HTTPS * Authenticate consumer application by verifying client id, client secret and auth token. * Pass data to process API for further processing. |
| 2 | customer-svc-proc-v1 | This mule application will be responsible for –   * Request /response transformation (if required) * Error Handling |
| 3 | zurich-insurance-sys-v1 | This mule application will be responsible for –   * Send transformed backend compatible request to Zurich Insurance System. |
| 4 | core-insurance-sys-v1 | This mule application will be responsible for –   * Send transformed backend compatible request to Insurance Core System. |

# Non-Functional Requirements

## Security

### Client Id Enforcement

Only HTTPS requests specifying client application credentials that are already registered with APIs are allowed to invoke this API

### Token Verification (OKTA OAuth2.0)

The policy allows HTTPS requests only if the token provided is valid. Policy validated the token by connecting to an authorization server.

# Sequence Diagram

This section depicts the solution via a detailed sequence diagram-

### Quotation API

Diagram

Description automatically generated

### Insurance API

Diagram

Description automatically generated

# List of API endpoints involved in the integration.

|  |  |  |  |
| --- | --- | --- | --- |
| No | API | Description | Endpoint (example) |
| 1 | Quotation API | **Search Quotes API -**  This endpoint it used to search travel insurance options based on input values.  Why POST : Since this API is accepting sensitive user data, sending search params in request body. | **[POST]** **/v1/travel/insurance/quotes/search** |
| 2 | Quotation API | **Save Quotes API -**  This endpoint is used to save the selected insurance option | **[POST]** **/v1/travel/insurance/quotes** |
| 3 | Insurance API | **Issue Policy API -**  This endpoint is used to issue insurance policy | **[POST]** **/v1/travel/insurance/policies** |

# API Request and Response

|  |  |  |  |
| --- | --- | --- | --- |
| No | API Name | Request | Response |
| 1 | Search Quotes API |  |  |
| 2 | Save Quotes API |  | 200 :      400 : |
| 3 | Insurance API |  |  |

# Exception Management & Error Handling

### System Error

Mule throws a system error when an exception occurs at the system level and no Mule event is involved.

A system error handler manages exceptions that occur:

● During application startup

● When a connection to an external system fails

When a system error occurs, Mule sends an error notification to registered listeners, logs the error, and if the error is caused by a connection failure, executes a reconnection strategy.

System error handlers are not configurable in Mule, and are handled by runtime itself.

### Validation Error

In any application development scope, error handling has a very important role to play. A sound error handling strategy and its implementation increases the application’s reliability and provides the application user higher visibility and confidence with the application.

Errors are handled by implementing a common error handler where errors are caught, logged and cloudhub notifications are sent with contextual messages.

Sample Error Response Structure:

Graphical user interface, text, application, chat or text message

Description automatically generated

Please note the following

* **Transient behavior**: Server errors and are therefore mostly transient, hence the api invocation should be retried. E.G. 5xx
* **Permanent behavior**: Server errors and are therefore mostly permanent, hence the API invocation should not be retried. E.G. 4xx range (Except 408, 429)
* The exceptions are caught and error messages/events are constructed accordingly. For the application containing HTTP listeners, descriptive error messages are sent back to consumers.
* The error message shall contain correlation/transaction id. The id will be used to audit and troubleshoot the errors by the operation team.
* API runtime policy-errors that occur at the API layer will be separately handled in the policy enforcement endpoint (If the API has http listener). Eg. errors from policy violations such client ID enforcement, rate limiting etc

Question & Answers

1. How to name the API

**Answer**: please refer section : [List of applications that are involved in integration](#_List_of_applications) and [List of API endpoints involved in the integration.](#_List_of_API)

1. Type of HTTP verbs to be used in each scenario.

**Answer**: please refer section : [List of API endpoints involved in the integration.](#_List_of_API)

1. How to version API when there is a breaking change?

**Answer :**

API versioning is the version of API interface. MuleSoft uses RAML to define API interface.

Lets take an example of insurance API mentioned above,

API with v1

version : v1.0

/quote/search:

Post:

/quote/save:

Post:

--------------------------------------------

API with v2

version : v2.0

/quote/search:

Post:

/quote/save:

Post:

/quote/{id}:

Get:

If existing client is using version 1, they will never know the additional endpoint /quote/{id} to get quote based on id.

**Solution :**

1. Maintain API version {{major}}.{{minor}} at design centre and APIM level. It is also important to fetch latest RAML in mule application configured in API kit router and auto discovery components.
2. **Major change in API (breaking change) :** Include API version (v1) in app url during the deployment

Eg. Application will have 2 urls one with v1 and other one with v2

https://{{doamin}}/{{applictaionName}}/**{{version}}**/travel/insurance/quote/search

1. Design pattern of API

**Answer**: please refer section : [Technical Design](#_Technical_Design)

1. How should the JSON request and response look like?

**Answer**: please refer section : [API Request and Response](#_API_Request_and)

1. How the API should react when there is a validation error or system error?

**Answer**: please refer section : [Exception Management & Error Handling](#_Exception_Management_&)

1. What happen when the core system is offline?

**Answer**:

**Implementing Circuit breaker in mule apps**

When system is down, we should automatically stop sending messages to CORE system for certain period.

This can be achieved by implementing **circuit breaker** processors.