# The Mule Travel organization system case study

Mule Travel is a well-established, medium-sized, nationwide travel reservation provider.

Currently, all customer representatives create **travel** **reservations** for customers using an **enterprise legacy system**. Mule Travel mainly handles large corporate customers, so their reservation systems also incorporate business processes to track corporate compliance and enforce regulations, particularly for government accounts.

The legacy system allows the travel agents to create new reservations and modify existing reservations. However, Mule Travel has recently started using a **CRM** **system** as its **reservations management tool**. In the first phase of the project, the CRM system is going to manage only customer data related to travel reservations.

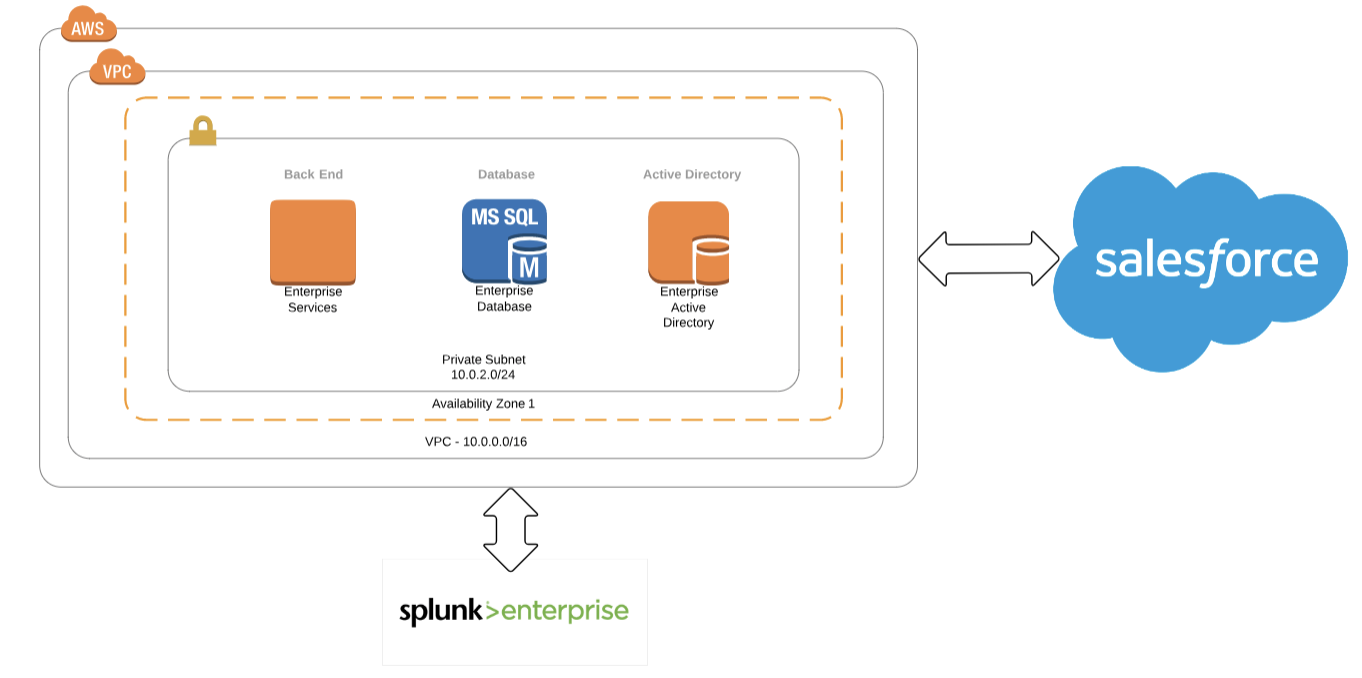
Currently, Mule Travel wants to maintain **customer information** in **both** the **legacy enterprise system** and the **CRM system**.

Therefore, it is important for Mule Travel **to synchronize customer information from the legacy enterprise system to the CRM system, and vice-versa**.

**Note: A key objective of the integration is to avoid point-to-point integration, but instead to design System APIs to these existing systems**

# Physical view of the Mule Travel organization's systems

This diagram summarizes the current physical view of the enterprise systems involved in the integration project.



# Kicking off the integration effort

Mule Travel’s plan is to initialize the integration effort by migrating all existing customer accounts and past and existing travel reservations from the legacy system to the CRM system in a one-time only load.

# A glimpse into Mule Travel's baseline technology architecture and functional requirements

### The existing legacy Java-based Enterprise System

Mule Travel’s legacy enterprise system is a Java-based monolith application. Salesforce is the **CRM system** used for **customer account and travel reservations management**. Mule Travel wants to implement this integration solution using MuleSoft's API-led Connectivity to design new Mule applications to allow travel agents to create new reservations and modify existing reservations.

### Planning for new System APIs

Mule Travel is planning to develop a **few System APIs** to access **customer information and reservations from Salesforce** and **reuse the existing enterprise legacy system**. Mule Travel will also break up the processing logic into System API, Process API, and Experience API layers. The enterprise wants to secure system APIs and wants to achieve low latency between the other API layers.

### Current enterprise analytics and monitoring

Currently, **enterprise monitoring and analytics** for all enterprise systems is provided by **Splunk**.

### Transactional monitoring is required

It is also required to be able to **trace transactions** across all the integration solution's API layers

### The existing legacy enterprise database system

Mule Travel has a legacy database system that is used as an **enterprise system** (monolith system). This enterprise system is not scalable, so Mule Travel wants to throttle request to this enterprise system. The enterprise system provides a SOAP service for communication.

The enterprise system (**ES**) is currently secured using AWS security. The Salesforce integration also has to be secured and use integration solution best practices.

### The existing legacy enterprise database system

The customer account information contains PII data such as SSN and address. The PII data should be encrypted in transit and also needs to be audited. If data is persisted in CloudHub, it must be secure and highly available. The logging and log retention policy for Mule applications should be clearly defined with alerts must be generated when a Mule application fails. Because these Mule applications are integrating critical path use cases, monitoring must also be setup for the Mule applications.

### Current environments in the organization

Mule Travel has **multiple environments** such as development, stage, QA, and production.

The **security** of all Mule applications and API implementations is a key concern for stakeholders, and they would like to have a clear demarcation between production and non-production environments. Mule Travel also wants to encrypt all passwords to external systems, use two-way SSL certificates with Salesforce, and host all its APIs within the MuleTravel.com domain.

### Strong security requirements

Mule Travel has an **identity management system** and wants to authenticate its Anypoint Platform users from this identity management system.

### Parallel work streams use Anypoint Platform

Mule Travel has multiple teams working on **Anypoint Platform**, and uses **GIT** as its **source code management system**. **Modularization** of the **Mule application** is highly encouraged at Mule Travel. **Reusability** of **libraries and APIs** and **automation** of **repetitive tasks** are equally important.

Mule Travel encourages team to create **reusable** **artifacts** and **promote** using **MuleSoft Anypoint Exchange** as **central repository** for artifacts.

The Mule Travel **performance team** is planning to conduct performance profiling of Mule applications in local and deployment sandbox, and is looking to implement **Continuous Integration and Continuous Delivery** (**CI/CD**) with Mule application configuration management. Configuration management must store and transmit all credentials securely.

**Non-functional requirements**

The customer data and travel reservations should be reliably transferred between the enterprise system and the Salesforce CRM system. Mule Travel requires a high availability **service level agreement** (**SLA**) of 99.99%. A load balancing or clustering solution can be considered to achieve high availability goals, but it must be cost effective.

The **response time** for the **API** should **not** be more than **300 milliseconds** and the **average** **throughput** from the API is **20 TPS**.

The communication between the **integration system**s should use **TLS**. It is important to select the right persistence for caches and batch jobs. **Data synchronization** between the **enterprise system** and the **Salesforce CRM** system is planned to maintain the integrity of both systems.

# Case study architecture

**Document views of the architecture for all stakeholders**

**Document all decisions made in the process of architecting**

**Document trade-off between meeting non-functional requirements and performance goals**

**Define the combination of deployment options for the Anypoint Platform control plane and runtime plane(s) which best serve this organization at the start of their strategic journey using MuleSoft**

**Define a network architecture that supports these requirement and deployment options**

**Define a CIDR block for an Anypoint Platform provided VPC that meets the organization requirement for a required range of private IP addresses**

**Define role-based access control (RBAC) to control and manage access to Anypoint Platform features**

**Define the most appropriate integration style for an integration solution with idiomatic usage of mule components for integration that meets the organization's current requirements**

**Plan the best way to implement the data transformation logic for this new Mule application while minimizing the overall testing effort**

**Plan the most appropriate way to implement persistence for the watermark in order to support the required data replication integration logic**

**Plan aspects of a CI/CD pipeline for Mule applications that can be automated using MuleSoft-provided Maven plugins**

**Plan an effective way to conduct performance tests of the API implementations within the performance environment**

**Define a logging strategy for Mule applications**

**Propose an architecture, design, runtimes, and OS/JVM/Network/Protocol tuning choices for integration solution**

**Plan an effective way for the project team responsible for the Mule applications and APIs being built to communicate with these stakeholders using Anypoint Platform and its supplied toolset**

**Plan and design a Mule application and related services so the credentials required to access all the related backend systems are managed centrally**

**Plan the type of Anypoint Exchange artifact(s) that should be added to Anypoint Exchange to publish a project skeleton to promote future reuse and parallel development**

**Plan the type of artifact(s) that should be added to the Mule application source files to properly publish the project source code to Anypoint Exchange**

**Plan to manage your libraries and shared resources in a Mule application**

**Define an effective alert and monitoring strategy that avoids infrastructure and application bottlenecks in the production environment**