<<Project Name>>

Solution Architecture Specification Document

MuleSoft®

Prepared by:	<<author name>>
Project Team:	<<collaborated team members>>
Version:	<<1.0>>
Date:	mm/dd/yyyy

Table of Contents

1 Document Control

1.1 Modification History:

Date	Version	Description	Author
<dd/mmm/yy>	<x.x>	<details>	<name>

1.2 Contributors:

Name	Email ID	Role

1.3 Reviewers:

Name	Email ID	Role

1.4 Approvers:

Name	Email ID	Role

1.5 Review Milestones:

Milestone	Date

2 Introduction

*[This section provides an overview of the entire **Architecture Specification Document**. It includes the purpose, scope, definitions, acronyms and abbreviations, overview and assumptions in describing the Architecture Specification of the given program]*

2.1 Purpose

*[This section defines the role or purpose of the **Architecture Specification Document**, in the overall project documentation, and briefly describes the structure of the document. The specific audiences for the document are identified, with an indication of how they are expected to use the document]*

2.2 Scope

[This section defines the details like what is affected or influenced by this document]

2.3 Definitions, Acronyms and Abbreviations

*[This section provides the definition of all terms, acronyms, and abbreviations required to properly interpret the **Architecture Specification Document**]*

Term	Definition

2.4 Overview

*[This section describes what and how the rest of the content is organized in **Architecture Specification Document**]*

2.5 Assumptions

*[This section lists out all the assumptions in consideration while writing this **Architecture Specification Document**]*

3 Requirements

[This section describes the requirements that must be met to achieve the objective, architects must ensure that the requirements are explicit, understandable, unambiguous and sufficiently complete for the purpose of conceptual specification]

3.1 Business requirements:

[Provide a high level, concise business level requirements view which are driving this initiative. Include a reference URL of the BRD, no need to recapture all the business requirements from BRD]

3.2 Technical requirements:

[Provide the list of high-level technical requirements such as functional, performance, usability, architectural, security, compliance and overall quality expectations]

3.3 Life cycle requirements:

[Based on the business and technical requirements, list down the solution life cycle requirements. Consider the factors such as process changes, scaling & adoption in future use perspective]

3.4 Functional requirements:

[Provide the list of processing functionalities needed for this solution. E.g. API requests, batch processing, custom workflows, data transformation etc. Describe what the system is required to do at detail level]

3.5 Compliance requirements:

[Provide the list of compliance standards or conventions must this implementation follow]

3.6 Performance requirements:

[What is the performance requirement at anticipated levels of scale for this solution]

3.7 Resiliency requirements:

[List down the resiliency requirements needed such as defining the criticality and SLAs to recover the system, data and operations as part of operational continuity and disaster recovery process]

3.8 Operational requirements:

[List down the monitoring and notification requirements granularity here. Define the SLAs for detecting the problems as part of support process. List down the administrative control details such as access to logs, restarts etc.]

4 Architecture

4.1.1 Reference architecture:

At MuleSoft, we encourage defining the solution architecture using the API led connectivity approach, since that would enable a given organization with fast and flexible integration model within their IT initiatives, let's take a look at the various components involved in this model:

4.1.1.1 System Layer

System APIs will provide a means of accessing the underlying/core systems (Salesforce, DB etc.) exposing that data in a canonical format, while providing downstream isolation from any interface changes or rationalization of those systems. These APIs will also change more infrequently and will be governed by Central IT given the importance of the underlying systems.

4.1.1.2 Process Layer

The underlying business processes that interact and shape this data should be strictly encapsulated independent of the source systems from which that data originates, as well as the target channels through which that data is to be delivered. These APIs perform specific business processes functions and provide access to non-central data and may be built by either Central IT or Line of Business IT.

4.1.1.3 Experience Layer

Data is now consumed across a broad set of channels/teams, each of which want access to the same data but in a variety of different forms. Experience APIs are the means by which data can be reconfigured so that it is most easily consumed by its intended audience, all from a common data source, rather than setting up separate point-to-point integrations for each channel.

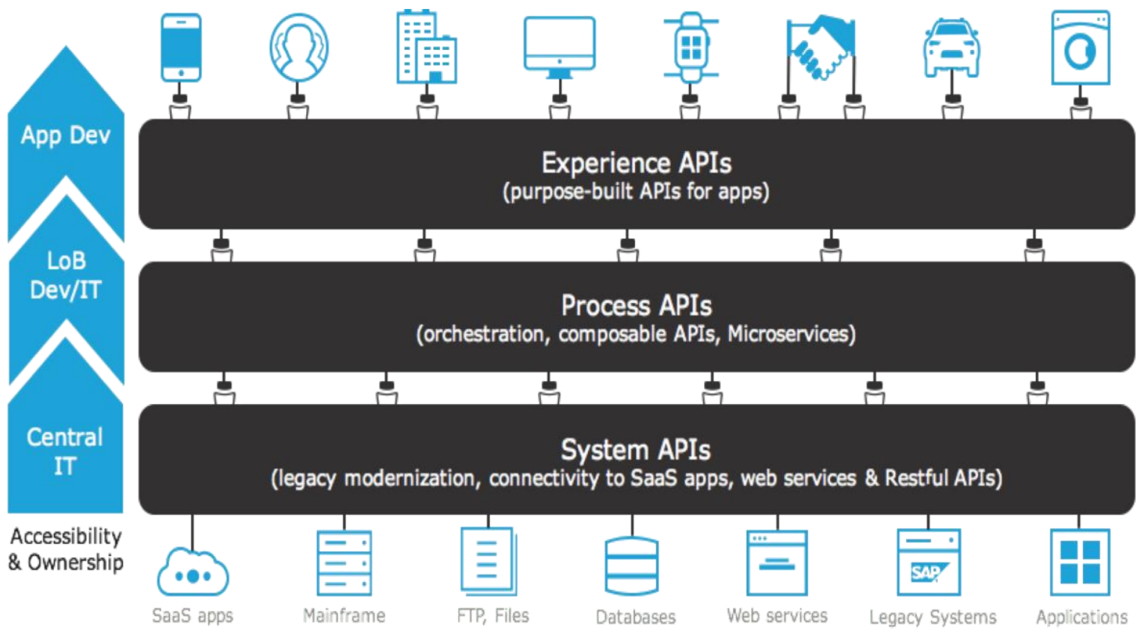
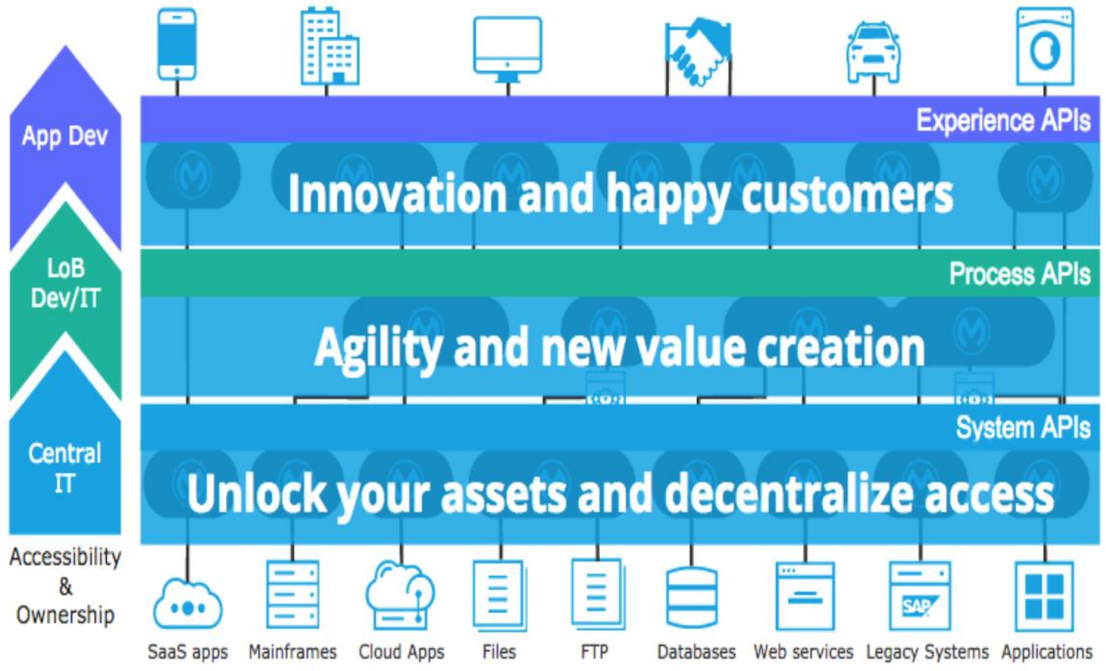

App Dev
LoB Dev/IT
Central IT
Accessibility & Ownership
Experience APIs
(purpose-built APIs for apps)
Process APIs
(orchestration, composable APIs, Microservices)
System APIs
(legacy modernization, connectivity to SaaS apps, web services & Restful APIs)
SaaS apps
Mainframe
FTP, Files
Databases
Web services
Legacy Systems
Applications

Figure 1 API led connectivity

The idea of applying this approach is to have scalable and reusable services, lower maintenance costs, faster time-to-market, flexibility, agility and a low learning curve.



4.1.2 Current architecture:

[Provide a conceptual drawing and narrative of the current solution, if applicable.]

4.1.3 Target architecture:

[Provide a conceptual drawing and narrative of the target solution (API led solution approach)]

4.1.4 Systems architecture:

[Provide a system level architectural diagram with a brief description of what each component in it helps achieve the overall solution]

4.2 Data

4.2.1 Logical Data Model:

[What logical data model this solution relies upon; mention details such as entities and the operations that are interrelated. Also add any additional documentation such as data flow diagram, sequential flow diagrams for each use case defined in the target architecture]

4.2.2 Systems of record:

[What are the systems that supplies or receives the data mentioned in above logical data model (e.g. Oracle, SAP, Salesforce, Active Directory). Mention the details of each system that supplies or receives the data]

4.2.3 Session / Transaction Management:

[How is state held and managed in the solution? Where is the processing stateless? How are transactions managed in the events such as commits/ rollback operations? How are transactions coordinated between multiple systems (Two phase commit etc.)]

4.3 Security

4.3.1 User security:

[Who are the users of this system? Are there any different types of users such as internal vs external and role-based access control rules defined? How many users will have access to this system? What about the authentication and authorization procedures? Please define (or) highlight the systems and flows that are responsible for authentication and authorization procedures.]

4.3.2 Data security:

[What key data is held in this system? Any PII (personal identifiable information) is captured as part of this solution implementation, operations and maintenance? Define the data classification level pertaining to the data maintained in this system, also mention the level of impact of data loss]

4.3.3 Application integration security:

[Describe the overall approach to the security of the solution and its components from transaction and infrastructure standpoint, additionally define the protocols, security policies, encryption models, authentication and authorization models considered for client app to API integration approach]

4.4 Performance

[How are the performance requirements stated earlier in requirements section are expected to be handled, how is performance measured at each component level, are there any out of box capabilities such as (logs, monitoring, platform analytics) planned to be used?]

4.5 Scalability

[What level of scalability can be achieved? Measure the proposed solution architecture and implementation by units such as userbase, data volumes etc. and mention the limits anticipated in this model. How is redundancy of the architecture is modeled to handle the scaling needs?]

4.6 Capacity

[What amount of capacity is required to support the scalability, performance and availability requirements defined? Measure and define the units such as CPU, memory, storage. Consider the capacity needs for administration, resiliency and monitoring (logs) purposes as well]

4.7 Operation

[What is the operational model for this solution? How is monitoring and alerting handled? What is the change management process?]

4.7.1 Activity/ status logging:

[Describe the log security, accessibility, compression, rotation and archival policies]

4.7.2 Error handling:

[Describe the error propagation and handling in system, notifications model, how each component recovers from error? Describe end user error experience (API responses, emails etc.)]

4.7.3 Change management:

[Describe the change management model, considering the aspects such as operational availability, any standardized process to be followed with respect to identify the schedules to apply the changes to solution? Describe any CI-CD dev-ops processes followed, any custom components used?]

5 Risks

[What are the various risks anticipating such as any security, technical etc. during the solution implementation]

6 Dependencies

[What are the various dependencies such as component, third party, cross functional team coordination etc.]

7 References

[List down any reference documentation that supports this architecture specification further]

- End of the document -