MakeSense

A Comprehensive Guide for Migrating from Mule 3.x to 4

Author: Srinivas Gadi, Head of Integration, MakeSense Pty Ltd

MuleSoft®

Contents

Introduction

Organisations have been investing on scaling their technology infrastructure finding ways to optimise/re-use their legacy systems which form the backbone running many core business operations. The earlier investments either has been in the form of adopting integration technologies or on custom-built integration solution patterns through cost compulsions or can be the mix of both.

Traditional architectural approaches centred around point-to-point integration creates huge bottlenecks for customer in their progress towards building a digital ready enterprise. These legacy integration solutions are holding back organisations from leveraging new digital technologies and creating new experiences for their customers and partners. Organisations need to adopt holistic approach leverages consultative practices helping their IT define and execute an efficient migration strategy that is built on maximising their existing investments and improving upon the existing architecture.

Mule upgrades made easier, simpler and faster

Customers have built robust solutions with sizeable investments in Mule 3.x. Approach for migration must be focused around maximising their investments and adopt prescriptive methods of analysing the current Mule architecture/implementations and follow the right recommendations. The right methods and best practices enable significant improvements and optimises the existing setup, cores and security implementations. These methods and best practices will help in choosing the right migration patterns and design constructs to significantly improve the efficiency and speed of the migrations.

Between its last major release 3.0 in 2010 and its current state, MuleSoft has evolved a lot, from being just an Integration/ API platform to something which can aid in assisting legacy modernisation, implementing secure SaaS integrations, and providing full API lifecycle management. MuleSoft now allows organisations to implement and allow its IT Department to integrate, connect and build its enterprise solutions in innovative new ways. The next-gen Mule 4 platform offers a broad range of new and improved features intended to enhance the capabilities of the platform along with developer experience. In this whitepaper, we try to address the benefits, challenges and best practices for organisations looking to migrate from Mule 3.0 version to Mule 4.x.

The 4.x version of the Mule provides enterprises superior and enhanced capabilities to make API/Integration implementations easier with improved performance and extensibility. Customers who moved to Mule 4.x are experiencing 60% efficiency in delivery and performance of the Mule applications. We strongly recommend customers to upgrade their existing MuleSoft 3.x API applications to Mule 4.x to benefit from all the improvements that brings in greater efficiency and performance improvement. MuleSoft 4.x Upgrade approach must include:

- Architecture validation and Rationalisation of existing Mule 3.x implementations
- Assess and follow the right Mule 4.x migration options and approaches
- Devise component by component migration approach for successful end-to-end Mule 4.x upgrade.

Glossary

Abbreviation	Definition
API	Application Programming Interface
IT	Information Technology
LOB	Line of Business
RAML	Restful API Modelling Language
SOA	Service Oriented Architecture

Approach and Methodologies for Mule 3 to Mule 4 Migration

The 4.x version of the Mule provides enterprises superior and enhanced capabilities to make API/Integration implementations easier with improved performance and extensibility. Customers who moved to Mule 4.x are experiencing 60% efficiency in delivery and performance of the Mule applications. We strongly recommend customers to upgrade their existing MuleSoft 3.x API applications to Mule 4.x to benefit from all the improvements that brings in greater efficiency and performance improvement.

While Mule 4 offers several new features that can make integration easy and cost-effective, migrating from Mule 3 to Mule 4 has its own challenges. To assist businesses in secured migration, we have detailed a few measures that you need to take prior to the migration process.

MuleSoft 4.x Upgrade approach must include:

- Chose migration options: As-is migration vs Rationalisation of existing Mule 3.x implementations
- Assess and follow the right Mule 4.x migration options and approaches
- Devise component by component migration approach for successful end-to-end Mule 4.x upgrade

Choosing Appropriate Migration Options

Organisations have different IT delivery structures based on their size and technology maturity. Certain enterprises have centralised IT with major platform decisions, architecture, strategy and platform innovation are driven centrally and individual LOBs would follow their recommendations and guidelines to build their API/integration solutions.

While in de-centralised enterprises, matured independent LOB's may have specific implementations with different architecture flavors, leading to multiple patterns and localised frameworks.

During the migration, customers should look at addressing these concerns in addition to looking at deployment architecture, HA, DevOps, etc. First things first, evaluate your business needs and road maps and consider the options of as–is migration vs re-architecture. The recommended approach though is mostly to re-architecture (based on factors such as cost, time, benefits, roadmap etc.).

For most organisations, the existing Mule 3 implementation would have matured over a period and could have been customised by multiple implementers. Thus, there is often a need to rationalise the existing integrations to support a smooth migration. Integration rationalisation can help organisations mature their integration landscape and improve LOB management capabilities and delivery of missioncritical business services. However, it requires buy-in from stakeholders across the enterprise - including senior leaders, technical teams, LOBs, enterprise teams, etc.

As-Is Migration

This strategy involves rebuilding the application as-is on the new platform. Here, implementation is done with none or very little modification in logic. While it is a simplistic approach to start with as-Is migration of APIs or implementation optimisation, this approach ensures to be quick and easy, as outcomes are achieved with minimal application disruption and effort. However, the platform may not deliver the latest features and benefits, which can lead to core optimisation. Note that organisations may leverage the MuleSoft-provided migration utility for the same too, which can help to a certain extent.

Re-architecture

This strategy entails making major changes in the integration implementation. This is a complex approach in comparison to as-is migration. We need to ensure that there is no impact on external behaviour of the middleware layer while implementing changes.

For example, let us consider that different LOB's have implemented customer information API's. As part of re-architecting, if the organisation decides to move to API-led connectivity with experience layers for each LOB along with few best practices and common frameworks embedded in the implementation, it may cause a few blips in the experience. Hence, testing is mandatory and has to be conducted end-to-end.

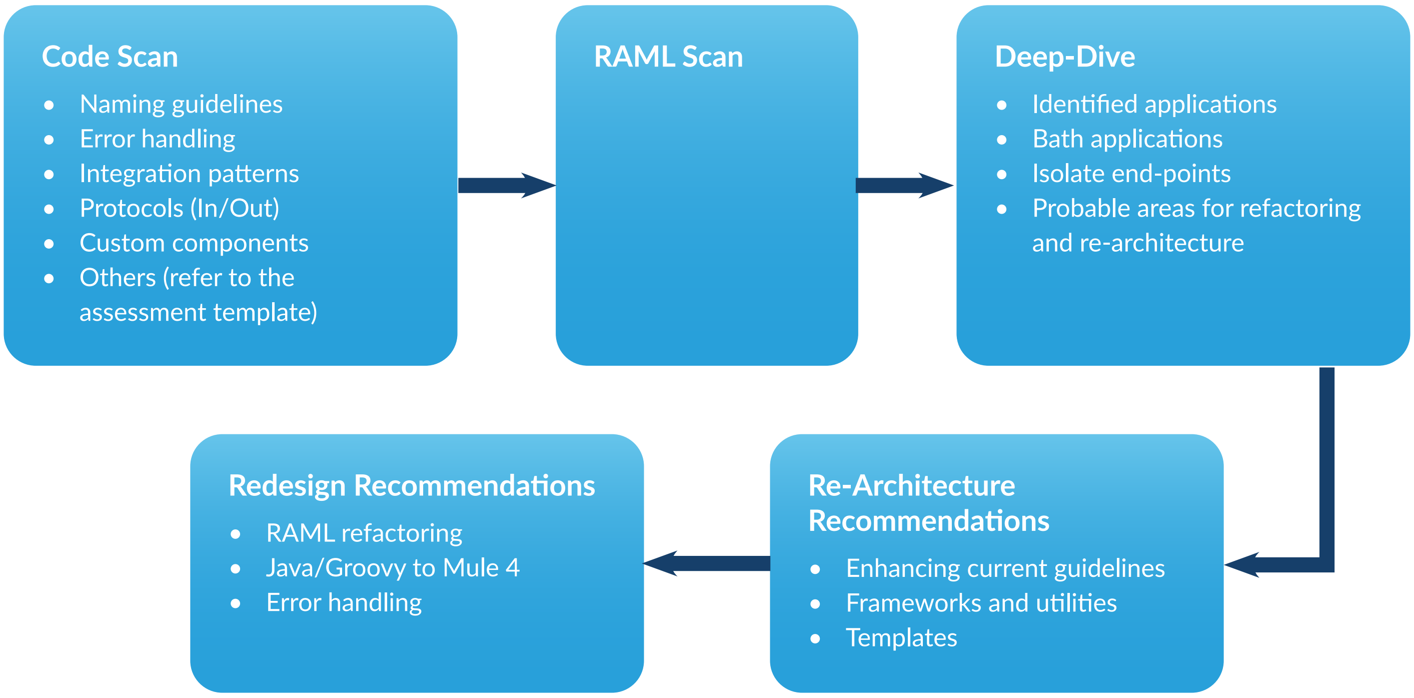
The organisation must choose between as-is or re-architecture, depending on the following factors.

Options	As-Is	Re-Architecture
Cost	Low	High
Time	Low	High
Version Compliance	Yes	No
Platform Benefits	No	Yes
Core Optimisation	No	Yes
Long-Term Benefits	No	Yes
Short-Term Plan	Yes	No

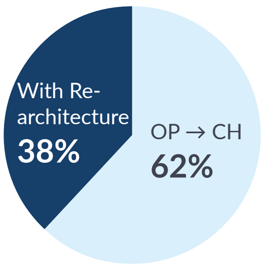
Based on our experience working on Mule migration projects and the objectives achieved, the comparison of effort vs. objectives must be assessed to select the right migration option.

Mule Migration – Start with Code Scan

Migration is a very complex process, specifically when the organisation has different standards and practices. The organisation not only has to rewrite/optimise the existing implementations, but also needs to ensure that this opportunity that requires significant investment is not wasted. We recommend the following approach to the key components starting with a code scan.



The following diagram gives an example depiction of assessment of each individual component during the code scan process.



CLOUDHUB MIGRATION

125 apps
- CloudHub – 29 applications
- On-premise – 96 applications

68 of 96 applications can be moved to CloudHub without re-architecture.

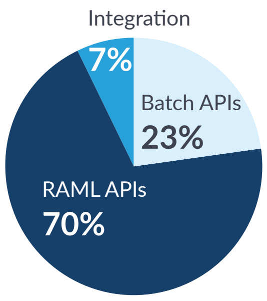
28 of 96 applications require re-architecture to migrate them to CloudHub.

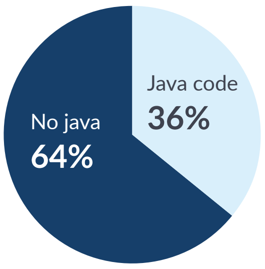
APP TYPES

29 apps are Batch/Data Sync APIs
- 3 apps use Batch components
- 27 apps use Flows with For each
- All are driven by Control –M
- Contains RAML interfaces

87 apps are Synchronous APIS.

9 apps are Integrations
- Doesn't contain RAML interfaces.





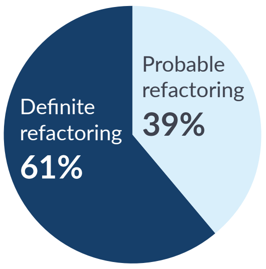
JAVA CODE

45 apps contain custom Java code
- File handling components
- Security components
- Base64 encoding components
- Some Groovy code.

RAML REFACTORING

Almost every app has a RAML interface definition, however:
- RAML definitions are aligned to REST
- No re-use through types or traits.

71 applications need RAML refactoring.
46 applications may need refactoring case-by-case basis.



Assessing Component by Component

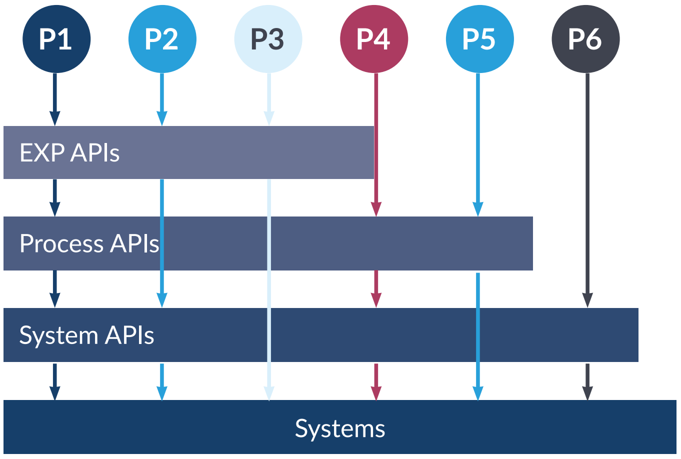
Assessing component by component provides opportunities to relook at implementation from various fronts and questions future viability. A few aspects to consider are:

Options	As-Is
Platform	Is the API hosted on-prem or on cloud? Can it be hosted on cloud? If not, why?
API-led	Is the API/ integration designed as per API-led thought process? Can they be relooked again, as some organisations would have over-engineered the implementation and now could relook and optimise?
RAML	Is there a refactoring of RAML required? Do we increase readability, implements types?
Custom	Are there custom Java implementations for file handling, parsing, reading messages from queues, backend applications etc.? Are there Groovy Scripts? Are there any customer components used?
Naming Conventions	Are there naming standards and are the implementations following them? Did they evolve over time and need standardisation?
Logging & Exception Handing	Are the standards and formats followed as per guidelines? Is it API kit default and do you want to relook?
End Connectors	Are there any system-specific connectors? Are there any object stores used?
Reusability	Is there any scope for reusability? Are there any common processes across LOBs?
Patterns	Have you identified the patterns applicable for your organisation? Are the implementations following those patterns?
Auto Discovery	Is auto-discovery implemented?
General Optimisations	General conditions, hard coding, certificates, removal of custom implementations with latest platform capabilities, optimise variable usage, conversion optimisation
Compliance	Does the implementation need to be compliant with PCI etc.?
Security	Is the security implemented across layers as per standards? Can someone hack if they pass through the DMZ?

Establish Patterns

Modern architecture sets a platform to support the future road map and should therefore be a key consideration while migrating. A well-planned and patterned integration migration will result in a modern platform that meets business needs for agility along with objectives and produces cost savings while aligning to the product roadmap.

One of the considerations as part of the migration is API-led connectivity and the identification of patterns for implementation. We believe there are generally six patterns in a synchronous world.



- Patterns P1, P2 aligns to the API-led approach
- Pattern P3 is proxy an external API
  - Connecting using native protocols in P3 is an anti pattern
- Pattern P4 is valid unless process needs a tight security
- Pattern P5 is not recommended unless
  - Requirement needs a complex transactional flow
  - Exception is obtained from Governance team
- Pattern P6 is valid for internal consumption unless system needs a tight security.

Separate the Functional Groups

Before starting the migration, it is better to prepare and start with foundational work like grouping and migrating common APIs, utilities, frameworks and reusable code. Prepare to migrate APIs, grouping similar APIs together will simplify development and ensure you have your foundations in place before you move to other apps.

Start with common APIs, like those associated with logging and infrastructure. Depending on the runtime deployment platform and subscription level, we may be able to leverage the new Anypoint Monitoring logging feature. Otherwise, we may want to consider sending your logging information to Splunk or the ELK stack.

After the foundational grouping and migration, next is to move on to domain-based migration. Logical groupings that perform common business functions - like APIs that handle customer data or those that update account information - should be migrated together.

Leveraging New Features of Mule 4

Migrating to Mule 4 from Mule 3 is a re-platforming exercise. While this brings in challenges, it also means there are new features that we should understand and review as part of the migration process. For instance, Anypoint Monitoring, initially introduced in 2018, has significantly changed from its previous version. The new version includes updated dashboard features, supports on-prem, hybrid, and CloudHub apps, and the ability to create alerts for app and server metrics. Among the new tools available in Mule 4 is Visualiser. This fantastic new tool lets you visualise the path of APIs from experience through process and system.

If the out-of-the-box API Management policies doesn't suit the needs, we can now create Custom API Management policies. Along with this, the Mule 3 Anypoint Enterprise Security Module (AES) has been split into different modules. We will need to review this fragmentation's impact on how it affects our migration to Mule 4.

One of the most significant feature changes is the move from DevKit to SDK. Connectors that were previously built in DevKit will need to be converted for Mule 4. Analogous to the MMA, but separate from it, there is a DevKit Migration Tool (DMT) available to help us re-factor your custom Connectors to run in the Mule 4 environment.

Optimise CI/CD Process

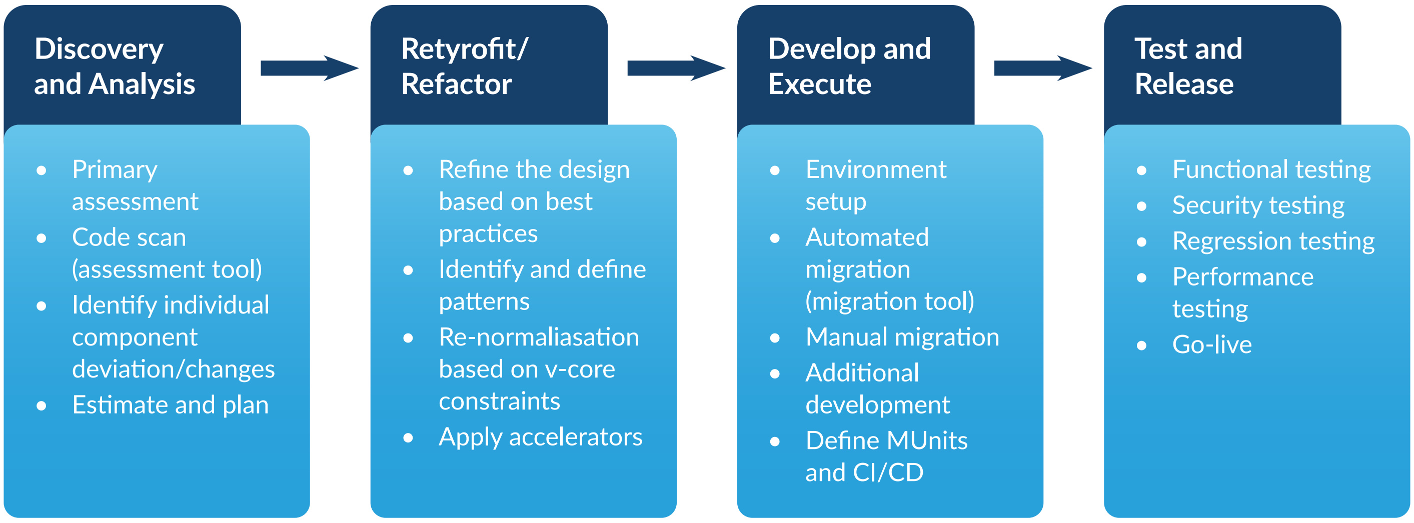
CI/CD offer process efficient way of automating the end-to-end development, testing and deployment process. If it is not currently available/leveraged, it is important that customers must leverage the superior capabilities that is provided by Mule 4 to improve and Fastrack the development process. Mule 4 provides improved support for Maven. Development, testing and automation teams should now look closely at automating their API build and deployment processes to their lower environments. While CI/CD may not apply to your enterprise, the benefits of an automated CI/CD process for non-prod environments is worth the design and planning time to implement.

Migration Implementation Process

We suggest organisations to take the below 4-step integrated approach before decommissioning the existing applications.

Better way to start the migration is to check the inventory on how many APIs that we have to migrate, and how complicated they are. Based on that insight, we will get a big picture of how much work we will have to spend on migration. Secondly, migration can be planned based on platform audit. Verify how many consumers use each of the deployed APIs. It will help us to reduce the work on unnecessary APIs and clean up the service repository. The number of consumers and API usage will also provide us with the information on which APIs to migrate first – this way we do not need to move the whole platform at once. It gives an opportunity to divide the initiative into smaller, agile-driven projects, with Mule 3 and Mule 4 working in parallel at the beginning. We can start with small, no-risk APIs, and then incrementally move APIs one by one to the new Mule 4 Platform.

If the client is running applications on premises, Mule 4 migration can be a good trigger to go cloud with some APIs, especially ones which require high availability and scalability. MuleSoft provides CloudHub platform, where we can deploy and manage Mule applications easily. It takes care of providing server provisioning and making clusters, and helps us to build end-to-end DevOps pipeline, using REST API for delivery processes.



Migration Sequence

Based on your needs and time available, if you need to migrate as-is from Mule 3 to Mule 4, it requires all the modules to be added in the Anypoint Studio palette. The sequence to follow when migrating is as follows:

Migrate Patterns

- Migrate Message Properties
- Migration Re-connection Strategies

Migrate Secure Property Placeholders

Migrate Watermarks

Migration Core Components

- Migrate Batch Components
- Migrate Choice Router
- Migrate Exception Strategies to Error Handlers
- Migrate Enrichers to Target Parameters
- Migrate Filters
- Migrate the 'For Each' Component
- Migrate Poll Component
- Migrate Scatter-Gather Router
- Migrate Transformers

Migration Connectors

- Migrate Anypoint Enterprise Security (AES)
- Module
- Migrate to the AMQP Connector
- Migrate Database Connector
- Migrate Email Connector
- Migrate File Connector
- Migrate FTP and SFTP Connector
- Migrate HTTP Connector
- Migrate JMS Connector
- Migrate Object Store Connector
- Migrate Scripting Module
- Migrate Spring Module
- Migrate Validate Module
- Migrate VM Module
- Migrate Web Service Consumer Module
- Migrate XML Module

MuleSoft Connector Migration from 3.x to 4.x

In Mule 4 the architecture for MuleSoft connector has changed completely. Connectors are developed in Mule 3 using Dev Kit, whereas in Mule 4 they are built using Mule 4 SDK. Each processor built, is added as a different connector (Pallet) in Anypoint Studio. To migrate the connector to Mule 4, you can use the conversion tool to:

To migrate the connector to Mule 4, you can use the conversion tool to:

- In pom file Enable Connector as Mule 4 extension
- Modify package element: <packaging>mule-extension</packaging>
- Modify parent element
  <parent>
  <groupId>org.mule.extensions</groupId>
- <artifactId>mule-modules-parent</artifactId>
  <version>1.0.0</version>
  </parent>
- Modify the folder structure. The typical folder structure for Mule 4 is:
  <Module>/api
  <Module>/internal
- Update annotations and Params Classes
- Delete/Add/Update Classes based on Mule 4 SDK (Please refer Java Docs).

Additionally, the following parameters need to be considered for Mule 4 migration:

- Auto Discovery
- DataWeave Header Content
- Standardise RAML Template.

Leverage Testing and DevOps framework

Most of the Mule 3.X implementation would have DevOps and Testing frameworks in place. Few of the good things that can be reused in a MuleSoft 3.X to 4.X migration are:

- DevOps frameworks with minor enhancements
- Testing framework, if built using tools (not Munit).

DevOps implementation will need changes in few places like the polling directory, project structures, scripts, while the configuration for the DevOps pipelines may change. Probably this could be a good time to see if there are any changes needed in the overall pipeline and approval process aligning to the organisation roadmap.

Automated testing if developed using MUnit cannot be reused. Only if the test cases are developed using tools such as Postman or SOAPUI projects (assuming that the API contracts/ Definitions/ Schema have not changed) it can be reused.

Conclusion

While Mule 4 Migration is a very complex process, specifically when the organisation has different standards and practices. We recommend to include architecture re-assessment to leverage the migration opportunity and refactor the existing applications to make best use of Mule 4 capabilities.

Though Mule 4 provides migration tools they only aid a little and needs significant changes to the code base to structures, patterns and components. With MakeSense expertise of successfully delivering large scale migration programs, customers can benefit with our proven experience in the form of templates, best practices and assets thereby significantly accelerating their migration initiatives leveraging the best of Mule 4 capabilities.

Have a question?
Need some advice?

Send us an email at contact@makesensesoft.com with the subject line ‘Migration from 3.x to 4.x’ or visit makesensesoft.com.

We would love to help you out.

MakeSense