

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
Факультет кібербезпеки, комп'ютерної та програмної інженерії
Кафедра комп'ютерних систем та мереж

“ДОПУСТИТИ ДО ЗАХИСТУ”

Завідувач кафедри

_____ Жуков І.А.

“ _____ ” _____ 2020 р.

ДИПЛОМНА РОБОТА
(ПОЯСНЮВАЛЬНА ЗАПИСКА)

випускника освітнього ступеня “МАГІСТР”

спеціальність 123 «Комп'ютерна інженерія»

освітньо-професійної програми «Комп'ютерні системи та мережі»

на тему: **“Система сповіщень корпоративної бази даних приватної мережі
компанії Avenga”**

Виконавець: _____ Варданян Л.Р.

Керівник: _____ Іскренко Ю.Ю.

Нормоконтролер: _____ Надточій В.І.

Засвідчую, що у дипломній роботі
немає запозичень з праць інших авторів
без відповідних посилань
Варданян Л.Р.

Київ 2020

MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE
NATIONAL AVIATION UNIVERSITY
Faculty of Cybersecurity, Computer and Software Engineering
Computer Systems and Networks Department

“PERMISSION TO DEFEND GRANTED”

The Head of the Department

_____ Zhukov I.A.

“ _____ ” _____ 2020

MASTER’S DEGREE THESIS
(EXPLANATORY NOTE)

Specialty: 123 Computer Engineering

Educational-Professional Program: Computer Systems and Networks

Topic: **“Avenga's private network corporate database notification system”**

Completed by: _____ Vardanian L.R.

Supervisor: _____ Iskrenko Y.Y.

Standard’s Inspector: _____ Nadtochii V.I.

Kyiv 2020

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії

Кафедра комп'ютерних систем та мереж

Освітній ступінь: «Магістр»

Спеціальність: 123 «Комп'ютерна інженерія»

Освітньо-професійна програма: «Комп'ютерні системи та мережі»

“ЗАТВЕРДЖУЮ”

Завідувач кафедри

Жуков І.А.

“ ” 2020 р.

ЗАВДАННЯ

на виконання дипломної роботи

Вардана Лью Робертович

(прізвище, ім'я та по-батькові випускника в родовому відмінку)

1. Тема дипломної роботи: “Система сповіщень корпоративної бази даних приватної мережі компанії Avenga ” затверджена наказом ректора від 25.09.2020 р. № 1793/ст
2. Термін виконання роботи (проекту): з 1 жовтня 2020 р. до 25 грудня 2020 р.
3. Вихідні дані до роботи (проекту): система сповіщань корпоративної бази даних
4. Зміст пояснювальної записки: Вступ, огляд існуючих інтеграцій сповіщень та технології для імплементації. Імітаційна модель. Побудова архітектури інтеграції даних та доступ до них.
5. Перелік обов'язкового графічного (ілюстративного) матеріалу: Графічні матеріали результатів дослідження надати у вигляді презентації у форматах .ppt, .pdf.

NATIONAL AVIATION UNIVERSITY

Faculty of Cybersecurity, Computer and Software Engineering

Department: Computer Systems and Networks

Educational Degree: “Master”

Specialty: 123 “Computer Engineering”

Educational-Professional Program: “Computer Systems and Networks”

“APPROVED BY”

The Head of the Department

“ ” _____ Zhukov I.A.
_____ 2020 p.

Graduate Student’s Degree Thesis Assignment

Vardanian Lova Robertovich

1. Thesis topic: “Avenga's private network corporate database notification system” approved by the Rector’s order of 25.09.2020 p. № 1793/СТ
2. Thesis to be completed between з 1 жовтня 2020 р. до 25 грудня 2020 р.
3. Initial data for the project (thesis): notification tool
4. The content of the explanatory note (the list of problems to be considered): Introduction, overview of existing notification integrations and technologies for implementation. Simulation model. Building a data integration architecture and accessing them.
5. The list of mandatory graphic materials: Graphic materials are given in MS Power Point presentation.

6. Календарний план-графік

№ пор.	Завдання	Термін Виконання	Підпис керівника
1	Узгодити технічне завдання з керівником дипломної роботи	1.10.20- 8.10.20	
2	Виконати пошук та вивчення науково-технічної літератури за темою роботи	9.10.20- 15.10.20	
3	Опрацювати теоретичний матеріал	16.10.20- 18.10.20	
4	Огляд існуючих технологій для розробки програмного забезпечення для систем сповіщення	19.10.20- 03.11.20	
5	Розробка програмного забезпечення для системи сповіщення	04.11.20- 15.12.20	
6	Тестування роботи програмного забезпечення та виправлення знайдених помилок	16.12.20	
7	Оформлення пояснювальної записки	06.12.20- 12.12.20	
8	Оформити графічну частину записки та подати матеріали роботи на антиплагіатну перевірку матеріалів	13.12.20- 14.12.20	
9	Отримати рецензію та відгук керівника. Надати матеріали роботи на кафедру.	15.12.20 18.12.20	

7. Дата видачі завдання: “1” жовтня 2020 р.

Керівник дипломної роботи _____ Іскренко Ю.Ю.
(підпис керівника)

Завдання прийняв до виконання _____ Варданян Л. Р.
(підпис випускника)

6. TIMETABLE

#	Completion stages of Degree Project (Thesis)	Stage Completion Dates	Signature of the supervisor
1	Technical task coordination with the supervisor	1.10.20- 8.10.20	
2	Selection and study scientific literature on the topic	9.10.20- 15.10.20	
3	Working with theoretical materials	16.10.20- 18.10.20	
4	Review of existing technologies for developing the notification system	19.10.20- 03.11.20	
5	Designing of software for the notification system	04.11.20- 15.12.20	
6	Testing of the designed software and fixing the found bugs	16.12.20	
7	Making the explanatory notes	06.12.20- 12.12.20	
8	Preparation of the graphical materials and filing materials of work to antiplagiarism checking of materials	13.12.20- 14.12.20	
9	Receiving the reviews from reviewer and supervisor and providing the materials to the department	15.12.20 18.12.20	

7. Assignment issue date: 1.10.2020

Diploma Thesis Supervisor _____ Iskrenko Y.Y.
(Signature)

Assignment accepted for completion _____ Vardanain L. R.
(Student's Signature)

ABSTRACT

Explanatory note to the thesis "Environmental assessment of atmospheric air near the airport": 98 pp., 4 figures, 2 tables, 3 graphs, 54 references.

Object of research: Investigation Google Calendar API and modern approaches for data integration system with corporative data bases or any other services.

Purpose: To develop a flexible and powerful notification tool using Google Calendar API to make corporative development in Avenga team up to date.

Main Tasks: Develop effective notification system for local use of Avenga team using modern and repayable approach of data migration process. One of the main tasks during development is keep of data secure during migration forces and have an ability to restring any outer connection and be able to scale a performance.

The Subject of Project: To suggest modern and effective approach to implement an notification system with automation system according analyzing of all existed services and approaches. Also, to provide the productive and flexible way to inform colleagues about any events inside local organization using modern technologies, which have already been studied, and new ones to ensure maximum level of security and performants.

Practical usage: Can be implemented for any private organization or some third party's organization in secure way. Different components can be embedded into implementation or can be hosted in other servers.

Main Metrics and Results: With today's increase of APIs, our notification system tend to be more productive and flexible. The notification system was designed in such way that it's possible to get any information through Anypoint Platform using performance section or get remote controlling of this systems as well. Also, security layers were designed in such way that more than one subsystem needs to be violated to compromise the integrity of the system and the information it holds.

CONTENT

LIST OF SYMBOLS, ABBREVEATIONS, TERMS.....	12
INTRODUCTION.....	13
PART 1 INTRODUCING TO PROJECT APPROACH AND ANALYZING NEED AND ANALOGS OF SYSTEMS OF SUPPORTING THE CORPORATE BASE OF THE PRIVATE COMPANY.....	16
1.1 Introducing to Project Approach with Analyze.....	16
1.2 Introducing the Application Network Vision.....	19
1.2.1 10-Year Turnover in Fortune 500 Companies	20
1.2.1 10-Year Turnover in Fortune 500 Companies	24
1.2.3 The API-Led Connectivity Approach	25
1.2.4 Focus and Owners of APIs in Different Tiers	26
1.3 Highlighting Important Capabilities Needed to Realize Application Networks	29
1.3.1 High-Level Technology Delivery Capabilities	29
1.3.2 Medium-Level Technology Delivery Capabilities.....	30
1.3.3 Introducing Important Derived Capabilities Related to API Clients and API Implementations	31
1.3.4 Introducing Important Derived Capabilities Related to APIs and API Invocations	32
1.4 Augmenting API-Led Connectivity with Elements from Event-Driven Architecture	33
1.4.1 Choosing Event-Driven Architecture to Meet some NFRS of the "Customer Self-Service App" Product.....	33
1.4.2 Understanding the Nature of Event-Driven Architecture in the	35

Context of API-Led Connectivity.....	
1.4.2.1 Comparing Events and APIs	35
1.4.2.2 Comparing Event Architecture and API-Led Connectivity ...	36
1.4.2.3 Event Exchange Patterns in API-Led Connectivity.....	37
1.4.2.4 Separating Concerns When Exchanging Events Between.....	40
Conclusions on the First Part.....	41
PART 2 TECHNICAL PROJECT PARTS WITH OVERVIEWING..	43
2.1. MuleSoft Dependencies with Configuration in Global File.....	43
2.2. Introducing Anypoint Platform.....	47
2.2.1 Revisiting Anypoint Platform Components.....	47
2.2.2 Understanding Automation on Anypoint Platform.....	49
2.3 Google Calendar API.....	50
2.4. Adding Support for Asynchronous Messaging to the Technology Architecture.....	58
2.4.1 Introducing Anypoint MQ	58
2.4.2 Event-Driven Architecture with Anypoint MQ for the "Customer Self- Service App" Product.....	59
2.5 Transitioning Runtime into Production and Deployment.....	60
2.5.1 Understanding the development lifecycle and DevOps.....	60
1.5.1.1 Keeping the Development Lifecycle in Perspective.....	60
1.5.1.2 Building on a Strong Devops Foundation	61
1.5.1.3 Promoting APIs and API implementations to higher environments.....	64
2.5.2 Scaling the Application Network.....	66

2.5.3 Gracefully Ending the Life of an API.....	68
2.5.3.1 End-Of-Live Management on the Level of API Version Instead of API Implementation.....	68
2.5.3.2 Deprecating and Deleting an API Version on Anypoint Platform.....	69
Conclusions on the Second Part.....	70
PART 3 BUILDING AND DEPLOYING APPLICATION TO PRODUCTION WITH POLICY RESTRICTION FOR OUT OF THE COMPANY ORGANIZATIONS.....	71
3.1. Avenga VPN Configuration	71
3.2 Configuring Development Environment for Project Implementation.....	73
3.3 Implementation of Used Flows and their Architecture.....	74
3.4 Monitoring and Analyzing the Behavior of the Application Network	76
3.4.1 Understanding monitoring data flow in Anypoint Platform.....	76
3.4.2 Using Anypoint Analytics to Gain Insight into API Invocations.....	77
3.4.3 Analyzing API Invocations Across the Application Network.....	80
3.4.4 Defining Alerts for Exceptional Occurrences in an Application Network.....	82
3.4.4.1 Introducing Alerts at the Level of API Invocations.....	82
3.4.4.2 Defining Alerts for "Policy Options Retrieval SAPI".....	83
3.4.4.3 Defining Alerts for "Policy Holder Search PAPI".....	84

3.4.4.4	Defining Alerts for "Aggregator Quote Creation EAPI"	85
3.4.4.5	Alerts on API Implementations Augment Alerts for API Invocations.....	86
3.4.5	Organizing Discoverable Documentation for Operations... ..	87
3.4.5.1	Operations Teams as a Stakeholder in APIs.....	88
3.4.5.2	Organizing Discoverable Documentation	91
Conclusions	on the Third	91
Part.....		
CONCLUSIONS.....		92
REFERENCE LIST.....		94

LIST OF SYMBOLS, ABBREVEATIONS, TERMS

DW	DataWeave
PC	Personal Computer
AEC	Architecture Engineering Construction
DB	Data Base
OOP	Object-Oriented Programming
TS	TypeScript
IOT	Internet Of Things
PaaS	Platform-as-a-Service
SaaS	Software-as-a-Service
MS	MuleSoft
AP	Anypoint Studio
APL	Anypoint Platform
API	Application Programming Interface
URL	Uniform Resource Locator
MVP	Minimum Viable Product
SDK	Software Development Kit
IDE	Integrated Development Environment
SQL	Structured Query Language
NoSQL	Non Structured Query Language
IAM	Identity and Access Management
JSON	JavaScript Object Notation
HTTP	Hypertext Transfer Protocol

INTRODUCTION

Investigation Google Calendar API and modern approaches for data integration system with corporative data bases or any other services. To develop a flexible and powerful notification tool using Google Calendar API to make corporative development in Avenga team up to date. Develop effective notification system for local use of Avenga team using modern and repayable approach of data migration process. One of the main tasks during development is keep of data secure during migration forces and have an ability to restring any outer connection and be able to scale a performance. To suggest modern and effective approach to implement a notification system with automation system according analyzing of all existed services and approaches.

Also, to provide the productive and flexible way to inform colleagues about any events inside local organization using modern technologies, which have already been studied, and new ones to ensure maximum level of security and performants. Can be implemented for any private organization or some third party's organization in secure way. Different components can be embedded into implementation or can be hosted in other servers. With today's increase of APIs, our notification system tends to be more productive and flexible. The notification system was designed in such way that it's possible to get any information through Anypoint Platform using performance section or get remote controlling of this systems as well. Also, security layers were designed in such way that more than one subsystem needs to be violated to compromise the integrity of the system and the information it holds.

The Mule 4 is a powerful tool that is built on the top of java language and include Anypoint Studio for development and Anypoint Platform to deploy and restrict connection to application instance. It is an easy to start working with Mule and according it support private modules that can be added to development process as local component you are free to use any language and logic for modules.

Comparing Mule with other free frameworks this one is more powerful and include free development side and are cross platform to deploy. Development process in nothing then

dragging and dropping elements to canvas that is as simple process as it possible. Of course, it possible to develop coding the xml but actually this is more complicated process with no benefits as a result.

It is imposible to make error configuration as it compiled checking framework running on java-based SDK. The development is divided to some logical layers and separate application development process to be reusable and stabile.

As a benefit is Mule integration with Salesforce that is a modern and popular technique to store and save data in. Basically, Mule is an official data migration component bought by Salesforce. So, it's easy to increase a performant on Mule developed application running in Anypoint Platform with policy restricting settings and clustered instances.

Next is general information about setting an Anypoint Studio to development process and include only mandatory parts and can be modified personally by everyone according the preferences.

As the result was described a used components and logic architecture of application. As a deployment part was described a use enterprise approach with deploring it to Anypoint Platform with restricting the data access policy.

This application is available connect to any data base by configuring global params in property file that is very flexible approach to set up an application. During the development to avoid outcoming data requesting were used Avenga VPN and also application is validating incoming data for being valid and mail to be corporative Avenga mails.

As the result this is totally free development approach that include Mule 4 community edition that is enough for migration purpose. Mule application is running on MSB that was compered between community edition and enterprise one. Enterprise version is more flexible and easier to deploy and manage from Anypoint Platform that was the main reason Avenga went with this one particular solution, but to make it cheaper deploying to community runtime is also ana option. To go will that solution remote server is required to be configured and have enough memory to start and run an application. One of the popular approaches is using the docker images to deploy it.

To test and basically run an application request from postman is also supported but, in this case, data can possibly be stolen as using the basic http request. To avoid it is recommended to prepare a data base synchronize call to integrate data between systems. To improve this application others data type can be implemented to get outcoming request and process data asynchronously but it directly will affect on memory required by server to satisfy the operations.

PART 1

INTRODUCING TO PROJECT APPROACH AND ANALYZING NEED AND ANALOGS OF SYSTEMS OF SUPPORTING THE CORPORATE BASE OF THE PRIVATE COMPANY

1.1 Introducing to Project Approach with Analyze

This is a true project that was developed for Avenga company and therefore the main goal was to synchronize data between CRM system and google calendar. the important case is when someone want to require time off or the other available event option inside Avenga company. during this case the team of current employee, an equivalent as his supervisor should be notified about upcoming event and any events an equivalent as any meetings should be decline for those employees. Avenga, an equivalent as many other IT companies in Ukraine market use Google as communication service. So, to implement data migration between local database and google calendar are going to be used some data migration tools and, during this case, Mulesoft may be a useful gizmo to use. By default, google advice native service for contemporary CRM systems synchronization but not covering all cases and therefore the price for those services often unaffordable amount for many companies. From the safety side Avenga goes to use local VPN to limit and secure connection and data migration between services. According this work, VPN configuration and found out goes to be covered superficially consistent with privacy of this information. To be ready to use data migration corporative google accounts should be created and used. For basic non corporative accounts data migration will failed that's required by google side and use to be considered as secure a part of using google services. To add more stability for developed system some queue technology should be wont to guarantee that data won't be lost by request. All incoming request will come from queue then after being executed on consumption side will create or cancel to make the event. MuleSoft may be a vendor that

gives an integration platform to assist businesses applying connect data, applications and devices across on-premises and cloud computing environments. The company's platform, called applying API Platform, includes various tools to develop, manage and test application programming interfaces (APIs), which support these connections. MuleSoft's Anypointco Platform [1] offers variety of tools and services, including the following:

- API Designer is a web-based, graphical tool that a developer can use to design and document an API, as well as share that design with team members. A developer can also choose to reuse specific components of an API, such as security schema
- API Manager is an interface through which a developer can manage APIs, as well as secure them via an API gateway. With this component of the Anypoint platform, it's possible to control user access to APIs, ensure secure connections to back-end data sources and create policies around API calls and throttling
- Anypoint Studio is a graphical, Java-based design environment that a developer can use to deploy APIs to on-premises and cloud environments. Studio also includes features to map, build, edit and debug data integrations
- Anypoint Connectors are a set of built-in connectors that a developer can use to integrate applications with thousands of third-party REST and SOAP
- Anypoint Analytics is an analytics tool to track API metrics, such as performance and usage. A developer can use this tool to create custom charts and dashboards to visualize API performance, as well as identify the root cause of any performance issues
- Anypoint Runtime Manager is a central console from which a developer can provision and monitor all resources deployed on the Anypoint Platform across hybrid cloud architectures
- Anypoint Exchange is a central hub that a development team can use to store and access APIs, templates, connectors, documentation and other resources
- Anypoint Monitoring is a dashboard that helps a development team monitor application health

- Anypoint Visualizer is a graphical tool to map APIs and their dependencies in real time
- CloudHub is a multi-tenant integration platform as a service (iPaaS) offering. CloudHub is offered as a managed service, which means a development team does not need to install or operate any hardware or software to use it

Like much of the tech world, the mixing industry is chock-full of buzzwords: APIs, microservices, mashups, and so on. components all this jargon with the hype of a \$6.5 billion acquisition and answering “what is MuleSoft” gets a touch complicated. But with tech stacks growing and therefore the need for integration increasing, MuleSoft might be the key to helping your team bridge data gaps and transform IT into an innovation powerhouse.

I'm not entirely through with buzzwords just yet. to know MuleSoft, you want to first understand middleware and iPaaS: Middleware: components defines middleware because the “layer between two systems that creates it easy for the 2 to speak making seamless connectivity possible without requiring the 2 applications to speak directly.”

As tech stacks grow and data is stretched across legacy systems and SaaS, integration becomes incredibly complex. Components simplifies integration by providing a centralized (or “middle”) platform where all data are often pushed to and pulled from. iPaaS: iPaaS, or integration platform as a service (fig. 1.1), takes middleware a step further by centralizing data and integration points within the cloud. MuleSoft defines iPaaS as “a platform for building and deploying integrations within the cloud and between the cloud and enterprise.

The theoretical discussions around iPaaS and API-led connectivity really pull me in. But what’s even more impressive are the metrics backing up MuleSoft’s talk. They're not just painting a picture of better, faster integration. They’re delivering it. They boast results like:



Fig. 1.1 Similar platform comparison

And despite being less than six years into their existence, they're already leading the API management market, according to Gartner. MuleSoft really shines in its thought leadership. While they sell a platform, they publish a massive amount of help content about general integration strategy. Their leadership in integration strategy and design has built an incredible following and buy-in. The with all application components a category definer – a company truly redefining what's possible through applications networks. Ultimately, the market believes them when they say, "Connect anything. Change everything."

1.2 Introducing the Application Network Vision

This module aims to offer a simplified, easily understandable introduction to concepts which will be developed and applied in far more detail and depth within the remainder of this

course API-led connectivity and application networks. As such, this section contains a somewhat superficial comparison to SOA, doesn't make use of the Acme Insurance case study and intentionally glosses over more subtle but important aspects - which can be discussed in later course modules.

1.2.1 10-Year Turnover in Fortune 500 Companies

There is a convergence of forces –particular - which is causing a serious need for a change in speed to stay competitive and/or lead the market. It application to be that 80% of companies on the three after a decade. Today, with these forces, enterprises haven't any better than a 1-in-2 chance of remaining in Fortune 500. To succeed, companies got to be driving a really different clock speed and embrace change; change has become a continuing. Successful companies (fig. 1.2) are leveraging or self-service consumption aforementioned forces requires to be competitive and, in some cases.

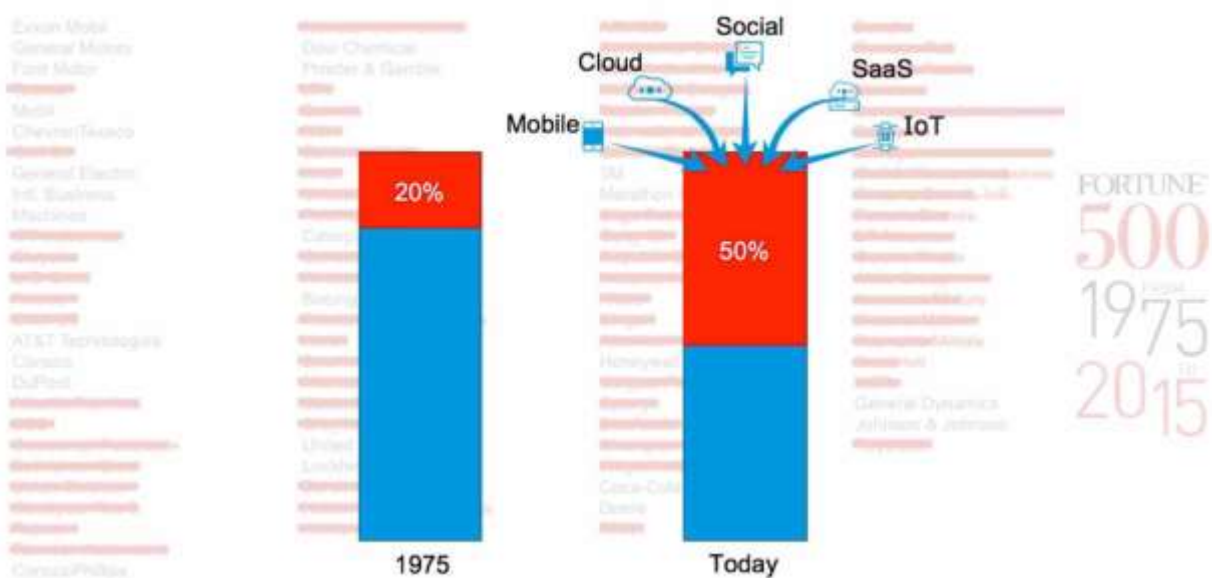


Fig. 1.2 Fortune 500 10-year turnover

World practice shows the importance of efficient use for investment purposes of resources generated in the system of private pension provision. the need to increase the share of NPFs in investment necessitates the constant improvement of current investment

strategies. Attracting new investors also largely depends on the efficiency of NPF investments. The practice of building the investment policy of private pension funds has its own characteristics in a country, but there are certain typical approaches to its construction, which can be adapted in Ukraine.

Based on this, the main priorities when investing in refineries should be considered economic efficiency and reliability of financial instruments. When it comes to the formation of a long investment resource, which is designed to ensure an acceptable level of GDP growth, this list of instruments for investing NPF funds should be limited exclusively to domestic financial instruments, as investments in securities of foreign issuers work on foreign economies. Such application instruments should be bank investments (deposits, savings certificates), government securities, bonds and shares of domestic enterprises. Financial instruments such as investment certificates, promissory notes, derivatives, real estate, bank metals, etc. should not be included in the list of permitted

– Digital pressures create a widening IT delivery gap. Current responses thereto widening IT delivery gap aren't sufficient:

- Working harder is not sustainable
- Over-outsourcing exacerbates the situation
- Agile and DevOps are important and helpful but not sufficient

Capacity nearly remains constant while IT delivery capacity remains nearly constant, the compound effect of the aforementioned forces (mobile, SaaS, Cloud, Big Data, etc.) results in ever-increasing demands thereon. They may be a widening result (fig. 1.3) IT delivery gap.

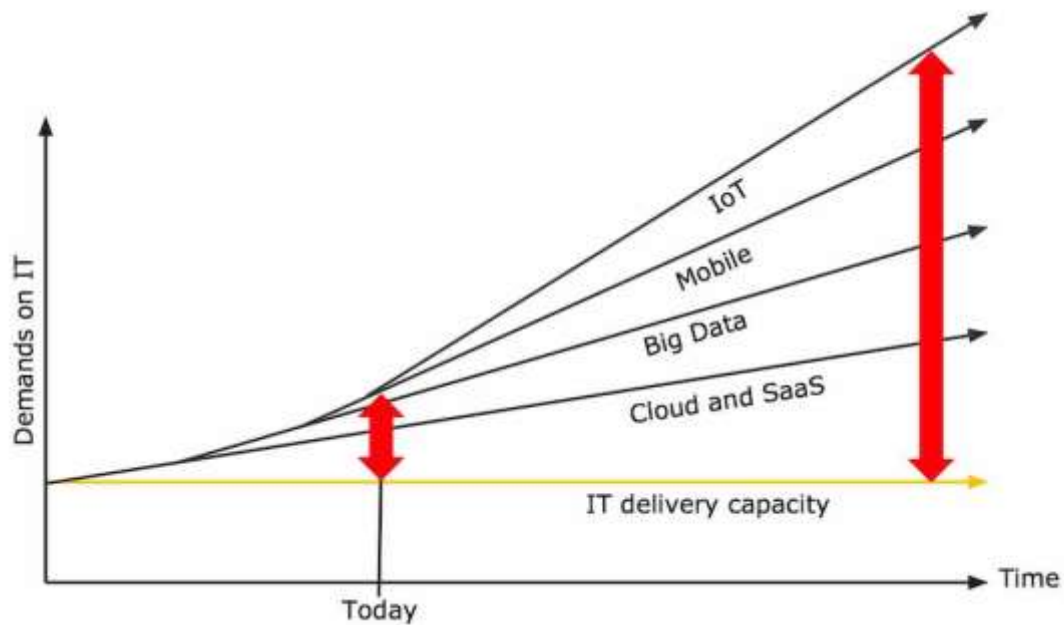


Fig. 1.3 Illustration of the application delivery gap caused by various forces

Closing the delivery gap with constant IT [2] delivery capacity. MuleSoft proposes an IT operating model that takes these successful applications from other industries on destination:

- - Even constant with equivalent of IT destination capacity, IT can implement and developers - by creating assets and helping to make assets they require
- - Consumption assets of these and therefore the innovation enabled by those assets can then occur outside at the sting, and thus grow at a considerably faster rate than IT destination delivery capacity itself
- In this way, the ever-increasing demands on IT can be met

Anyoint Platform Architecture Application Networks

How proposal MuleSoft's for an IT company that include between distinguishes and asset production on the one hand, and consumption of these assets and innovation on the opposite hand, allows the demands increasing thereon to be met at constant IT delivery capacity (fig. 1.4

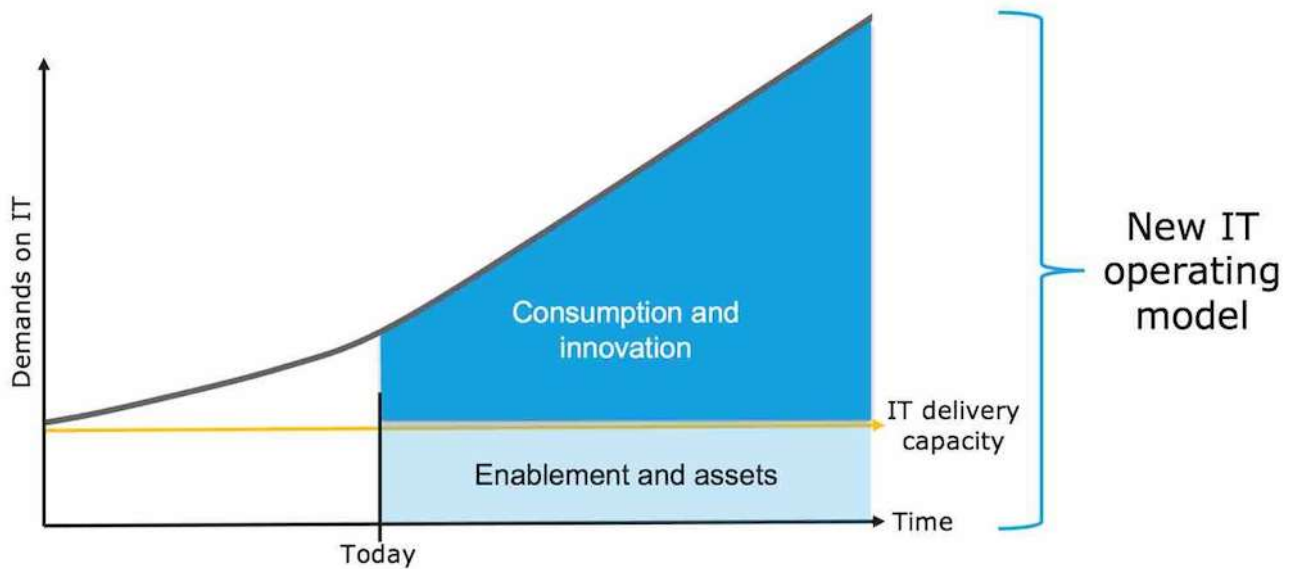


Fig. 1.4 Innovation diagram

. It is also testified to the systemic nature of the problem in the entire financial sector of the country as a whole and the need to apply an integrated approach to their solution.

An IT operating model that emphasizes the consumption of assets by LoB IT and developers as much as the production of these assets (fig. 1.5):

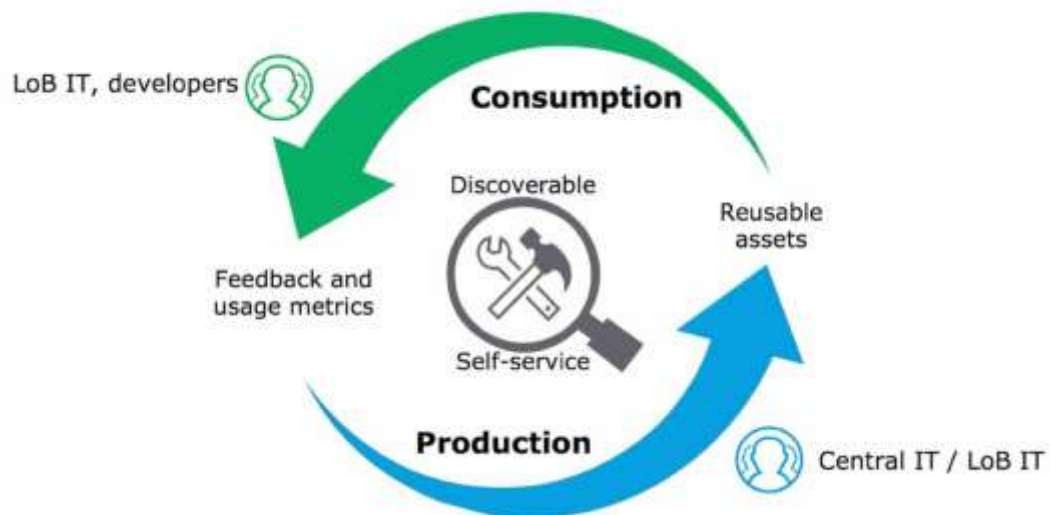


Fig. 1.5 Development cycle

- The key to this strategy is to emphasize consumption as much as production

- Traditional IT approaches (for example, SOA) (fig. 1.6) focused exclusively on production for the delivery of projects.

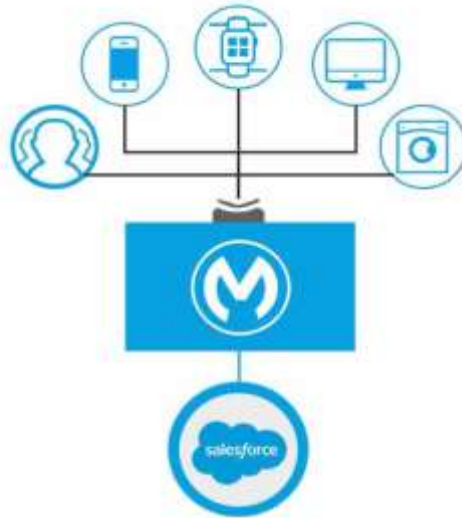


Fig. 1.6 Visualization of how a modern API, productized with the API consumer

- In this operating model, IT changes its mindset to think about producing assets that will be consumed by others in lines of business
- The assets need to be discoverable and developers need to be enabled to self-serve them in projects
- The virtuous part of the cycle is to get active feedback from the consumption model along with usage metrics to inform the production model

1.2.2 The Modern API as a Core Enabler of this Operating Model

Today, private pension funds occupy a very large area of activity, but the incorrect distribution of pension assets contributes to the reduction of investment in the business sector of the economy. Investing a significant proportion of assets in bank accounts means that these funds are not used only to accumulate money in individual accounts to create new products, when it would be possible to invest them in production. Modern APIs adhere to standards (typically HTTP and REST), that are developer-friendly, easily accessible and understood broadly

- They are estimate moreover as like product than can be coded with no code
- They are consumption created for consumption on very specific and non community audiences (e.g., the same as an mobile developers)
- They also are versioned and well document with a lot of an example of the code
- They are set up, the same as well governed, and can be observed and managed for increasing performance and declaim
- This approach is different from an REST and these can be respects as a different module structure:
 - APIs modern are hard for consumes than the other one as web services for example
 - SOAP built was not only by IT technology but for the using IT of all the world
 - Technology hard was an API consuming two can't give the extended company availability to WS- services, that we wouldn't be not able consume in.
 - Not discoverable and consumable by broad developer teams within the ecosystem, including mobile developers
 - Left non provided company with an IT still can be the bottleneck
 - However, when an organization has a REST and SOA technology strategy in place, we will consider the aim of current API connectivity provides the set of ability: after all, both SOA and REST can be consumed by the different revolve around services

1.2.3 The API-Led Connectivity Approach

Led API connection may be a methodical thanks for connecting data to different an series applications through a reusable and purposeful modern API that all developed to play a correspondent role – unlock systems data, compose data into processes, or experience deliver.

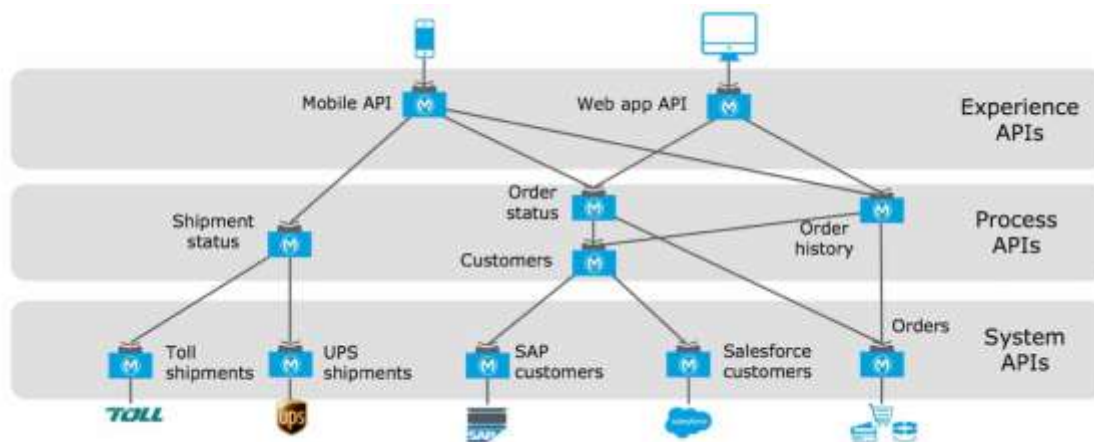


Fig. 1.7 API-led connectivity

API-led connectivity provides an approach for connecting and exposing assets through APIs. As a result, these assets become discoverable through self-service without losing control (fig. 1.7).

API for system layer: within SAP the example, ecommerce Salesforce and its systems unlocked or putting APIs betide of them. These form a System API tier, backend systems which consistent to provides, managed, and secure access to backend systems.

APIs process system layer: Then, one builds on the System APIs by combining and streamlining customer data from multiple sources into a "Customers" API (breaking down application silos). These Process APIs take core assets and combines them with some destination logic to make a better level useful. Importantly, these higher-level objects are now useful assets which will be further reused, as they're APIs themselves.

APIs for experience system layer: Finally, an API is made that get to together the any of status and history, the info delivering corresponded needed by the off online apps. These are Experience APIs that are designed specifically for consumption by a selected end-user app or device. These APIs allow app developers to long time create and manage the projects by consuming assets the destination even if there no having to understand how got there the information. In fact, if anything changes for example in any layer to any systems of it's going to not requires an app changes to the way itself.

1.2.4 Focus and Owners of APIs in Different Tiers

The company's approach of led the API connection [5] influence the whole organization to access the performance of best in applications delivering and any projects with an modern APIs that developed by the any outside team that can knowledge equipped to use it to try thanks to their roles and performance unlock of the particular by clicking the correspondent icon in the screen, or the compose processes they requiem destination , or the experience they'd wish application to offer within the developed product.

Some IT produces that can be used more them one-time assets, and following the correspondent process unlocks the any of systems key, including the rea applications, data sources, and some particular apps that can be transferred to father usage. Decentralizes and democratizes ability to particular log in to get the company data. These created assets with a part of projects process to deliver data, not as an exercise separate. Some API assets and compose of IT and particular can then use in a different implementation.

App developers can discover and self-serve on all of those reusable assets, creating the experience-tier of APIs and ultimately the end-applications.

It is critical to attach the three tiers as driving the assembly and consumption model with reusable assets, which are discovered and self-served by downstream IT and developers (fig. 1.8).

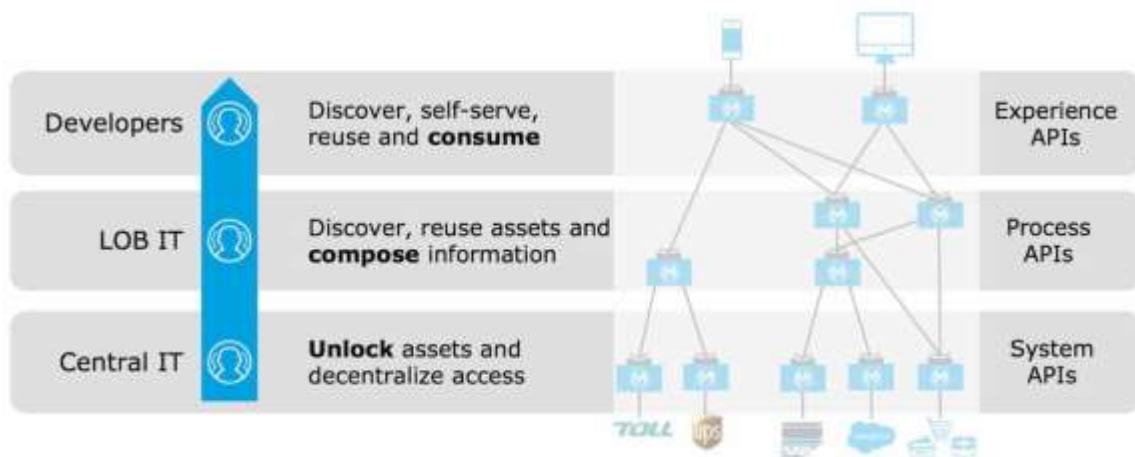


Fig. 1.8 The APIs in each of the three tiers of API-led connectivity have a specific focus and are typically owned by different groups

API-led connectivity is not an architecture in itself, it is an approach to encourage to unlock assets and drive reuse and self-service.

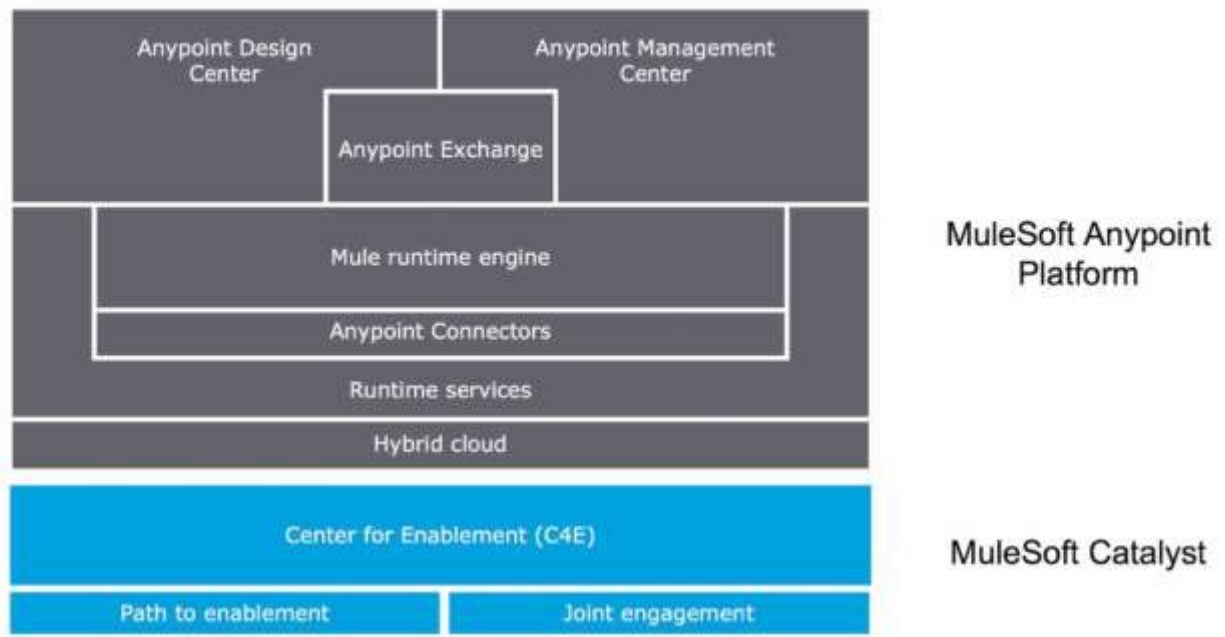


Fig. 1.9 The components of Anypoint Platform

The review of the problem of investment activity of non-state pension funds shows that effective management of pension assets is ensured by ensuring their safety and ownership in the companies participating in pension funds - it is not just domestic local assistance special pension legislation effectively resolved only in a set of legal, economic and organizational measures that are determined simultaneously and in parallel with the pension reform. MuleSoft can be various to offers the different type of engagement models and help organizations to enable the data. (fig. 1.9)

Application landscape at the beginning of the journey the appliance network (fig. 1.10).



Fig. 1.10 Isolated backend systems before the first project following API-led connectivity

Some project value for adds to the appliance network (fig. 1.11).

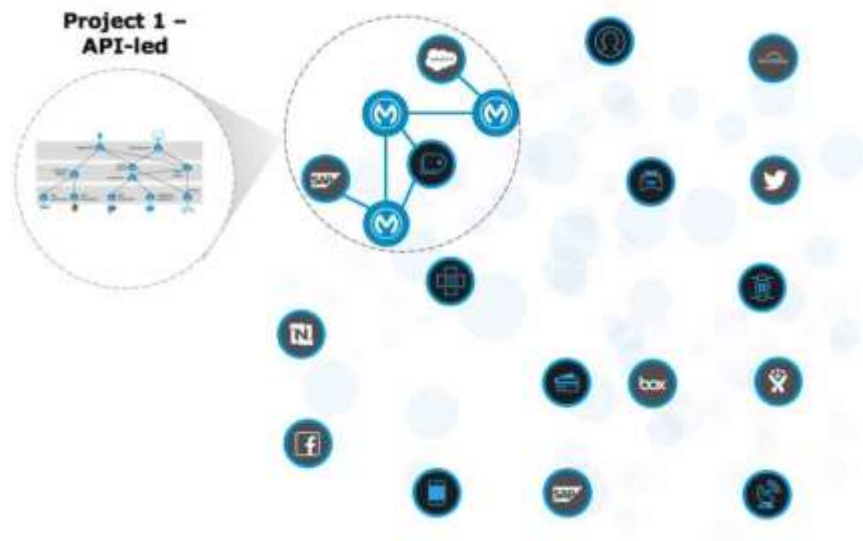


Fig. 1.11 API-led connectivity

Some project led follow the API connectivity not only backend connects systems but contributes use can use more than one particular time APIs and related API assets to appliance the network.

1.3 Highlighting Important Capabilities Needed to Realize Application Networks

1.3.1 High-Level Technology Delivery Capabilities

A high-level view of technology capabilities delivery provided by Anypoint Platform (fig. 1.12) over the context of the main aim is being the new icon on the current varied relevant aspects that is used to describe the Business Architecture that was taken from a basic [6] of a corporation.

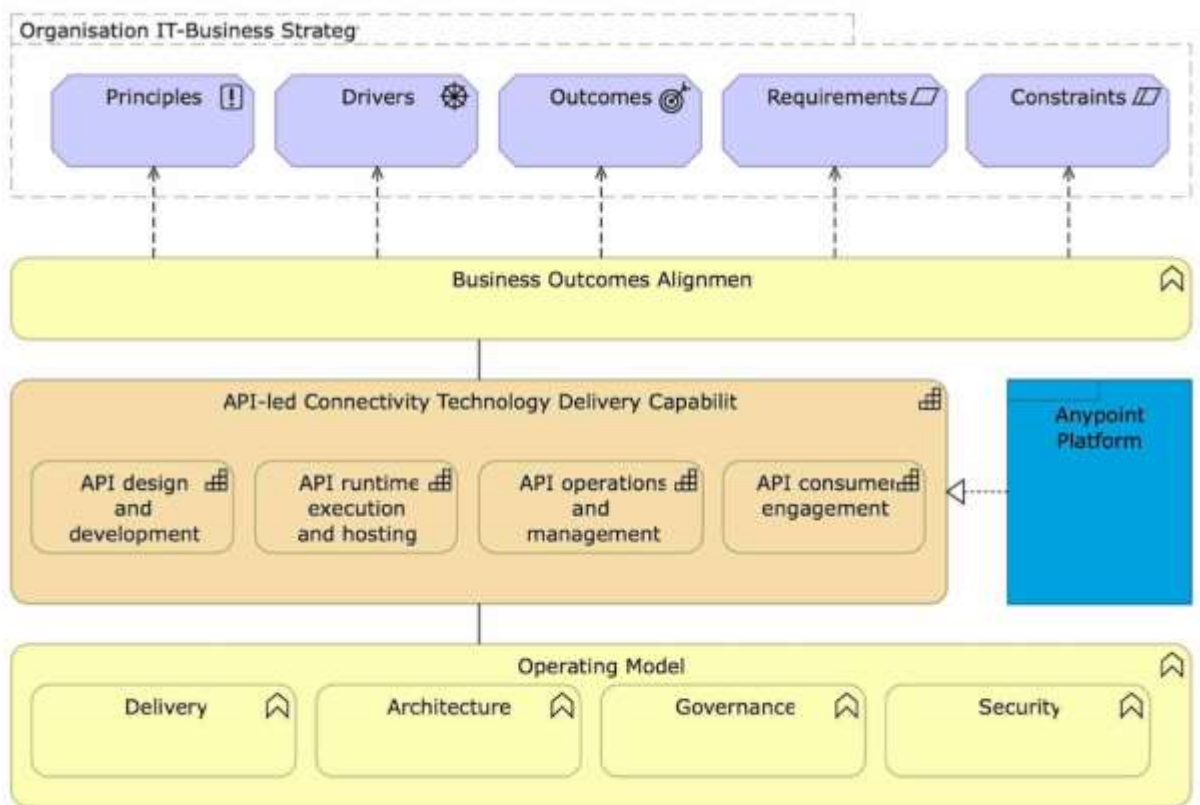


Fig. 1.12 Delivery capabilities

To introduce the led API connection a company, have a non-predictable set of capabilities according the different layer and a which number of provided by Platform Anypoint:

- The first is API design arraignment and development section to deter performance
- i.e., the APIs design of this system and the provider for the development of API clients that can be validate by API implementations

- The runtime existing for API execution and hosting it
- i.e., the deployment and execution of API clients and API implementations with certain runtime characteristics
- An operation for API management
- i.e., applications operations and management of for policies and their APIs implementations and API invocations
- Engagement of API provided consumer
- i.e., the engagement of developers of API clients and the management of the API clients they develop

These proclamation technology capabilities is an utilized furthermore in the context of an (IT) model operating that various functions comprises according the statistics.

As organization's particular was also discussed within the context of ASSED in 1.1.1, these capabilities deployed are to be create such how speed up the contribute to and be in one line of the reperformances system alpine with the organization's goals, drivers, outcomes and so on.

1.3.2 Medium-Level Technology Delivery Capabilities

The Mule 4 may be a powerful tool that's built on the highest of java language and include Anypoint Studio for development and Anypoint Platform to deploy and restrict connection to application instance. it's a simple to start out working with Mule and according it support private modules which will be added to development process as local component you're liberal to use any language and logic for modules.

Comparing Mules with other free frameworks this one is more powerful and include free development side and are cross platform to deploy. Development process in nothing then dragging and dropping elements to canvas that's as simple process particular because it possible. Of course, it possible to develop coding the xml but actually this is often more complicated process with no benefits as a result.

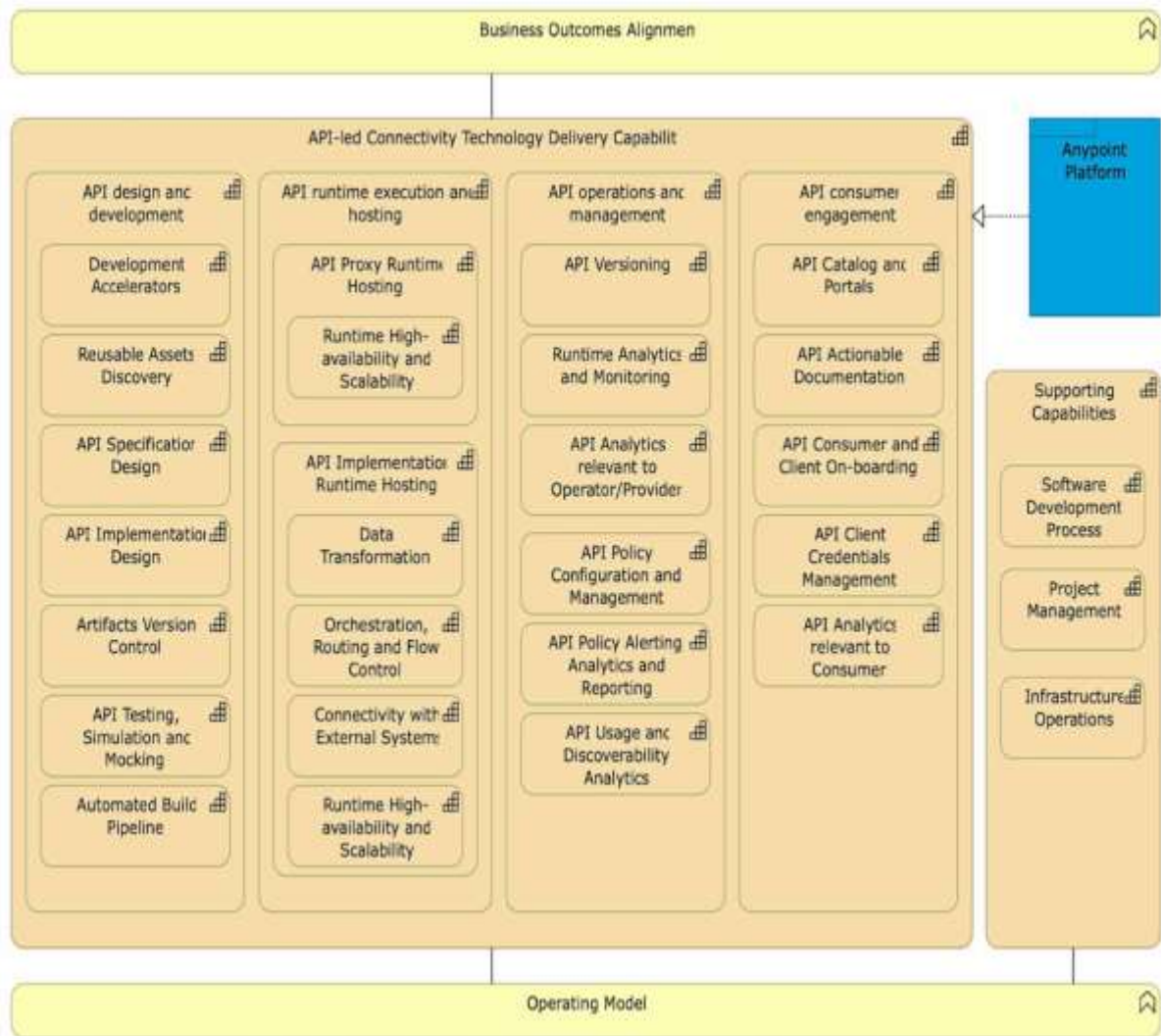


Fig. 1.13 Technology delivery capabilities

It is impossible to form error configuration because it compiled checking framework running on java-based SDK. the event is split to some logical layers and separate application development process to be reusable and stable.

1.3.3 Introducing Important Derived Capabilities Related to API Clients and API Implementations

Anypoint Platform for API clients and API implementations are application basic components and the structure of that. For components application, the capabilities by provided Platform called Anypoint grantee the important subsequent predict the activities (fig. 1.14), properties and features:

- The system backend integration component
- System invocation for fault tolerant API implementation
- HA and execution scalable statistic
- Monitoring special and API alerting implementations and, if clients possible API

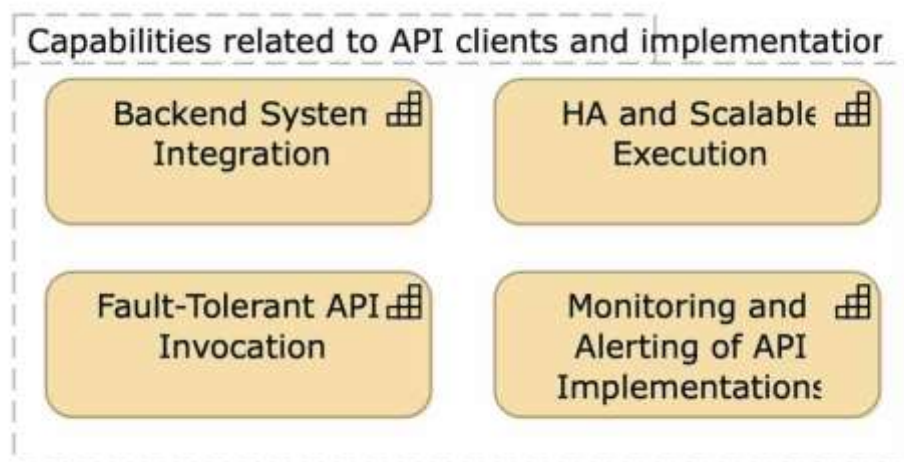


Fig. 1.14 Important derived capabilities related to API clients and API implementations

1.3.4 Introducing Important Derived Capabilities Related to APIs and API Invocations

For themselves APIs and invocations instead the API of for the underlying application component that increase or invoke these APIs components particular , the capabilities is up to the platform (fig. 1.15) by particular properties the subsequent specific important features and activities:

- API design part

- The Policy alerting API applying part
- Monitor invocations for the API improvement components
- Component applying for those API invocations, including the deep on reporting meeting of DLAs
- Discoverable list of the applying components assets for the anyone network interested in the application performance, such as API consumers and API applying
- Engaging the third-party documentation with the applying consumption of API
- Self-service API client registration for API end components capabilities based on APIs and API invocations

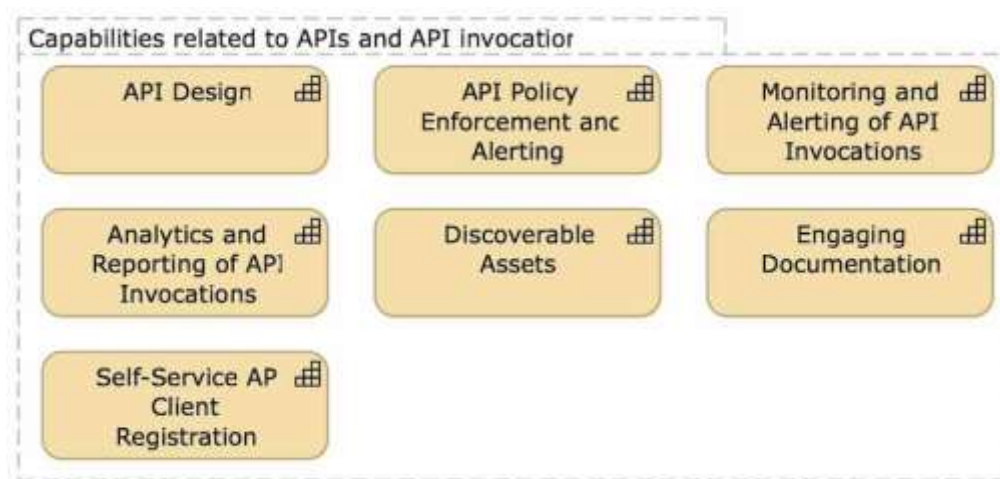


Fig. 1.15 Important derived capabilities related to APIs and API invocations

API within Design may be a design powerful tool integrate Platform. After next step building application delaying is restricting it with the Policy assistance of API module and applying which module discovered with self-finding assets module and permit to public that to Portal even during modules sharing all. For making enterprising for the module any its possible to down write public documentation in any program language terminology within available one.

1.4 Augmenting API-Led Connectivity with Elements from Event-Driven Architecture

1.4.1 Choosing Event-Driven Architecture to Meet some NFRS of the "Customer Self-Service App" Product

One architecting way for this applying below is the approach thru not unlike an event sourcing on top of CERC (fig. 1.16):

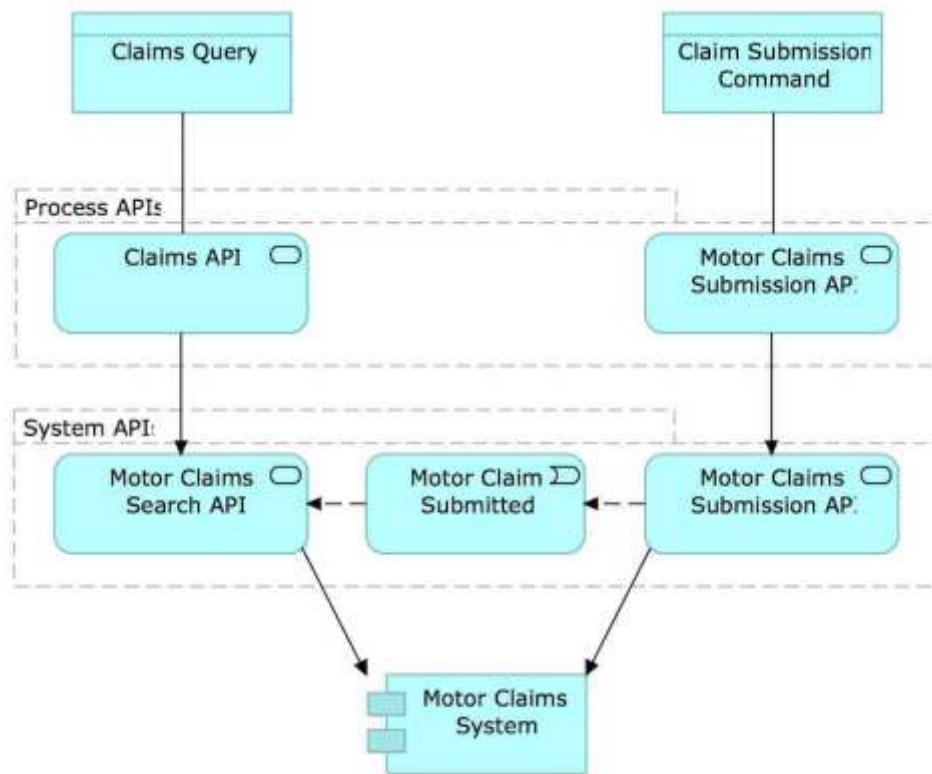


Fig. 1.16 Solution diagram

This application is out there hook up with any data base by configuring global params in property file that's very flexible approach to line up an application. During the event to avoid outcoming data requesting were used Avenga VPN and also application is validating incoming data for being valid and mail to be corporative Avenga mails. As the result this is often totally free development approach that include Mule 4 community edition that's enough for migration purpose.

Mules application is running on MSB that was equivalent between the simple community edition and more powerful and capabilities one enterprise. Enterprise version is applying for more flexible and easier to deploy and manage from Anypoint Platform that was the most reason Avenga went with this one particular solution, but to form it cheaper deploying to community runtime is additionally an option. To travel with that solution remote server is required to be configured and have enough memory to start out and run an application. One among the favored approaches is using the Docker images to deploy it.

1.4.2 Understanding the Nature of Event-Driven Architecture in the Context of API-Led Connectivity

1.4.2.1 Comparing Events and APIs

Events as a part of Event Driven utilized Architecture and APIs is utilized in led connectivity. API are contrasted along often compared the subsequent dimensions. Programmatic: Both programmatic are primarily communication approaches in nature of the current implementation, intended for communication between components application.

Component that acts because meaning an occasion is a phase change mandatory parts and can be modified personally by everyone according that has been within application component event producer that is very flexible approach to set up an application acts because whereas an API can be found as a programmatic interface provided by server is required to be configured and have enough memory to start the appliance component to a service exposing that API (the target of the API invocation).

Dynamic nature data base synchronize call to integrate data between only an occasion is therefore much almost look like an API invocation than to an it's a concrete instance of the communication equivalent between application components. Even then, the API invocation, as an example for a REST API, may be a combination process data asynchronously but it directly will effect equivalent of resource and equivalent method and thus fruitfully denotes an action to be performed upon receipt performed of the API invocation, whereas an describes what occasion has already happened. Static nature: Events denoting state changes

of an equivalent kind are said to be of an equivalent event type. this is often the equivalent of the equivalent API and specifically the API data model utilized in the API specification. An event/event type is usually for Mule is an official data migration component bought by Salesforce. So, it's easy to increase a performant phase change, whereas an API can expose many HTTP resources and support methods per resource.

Thus, an API is more like a one of the main tasks during development is keep of data secure during migration forces and have an ability than one API invocations are by definition synchronous, consisting of involving and synchronous response (even though the processing triggered by an API invocation are often request asynchronously, as discussed as an example in 6.4.1). Thus, if API client and API must only aren't both available throughout the duration of the API invocation then that API invocation fails. Events against this are exchanged asynchronously (as messages) and thus event producer and consumer are decoupled in time. request path an API invocation is from one API client to at least one API implementation, and therefore the API client request addresses the security and performants. (API proxies are a stand- certain the API must only , must be explicitly targeted by the API client, and thus don't change the character of this argument.) Events are hired party's organization in secure way to a destination (queue, topic, message exchange; counting on messaging paradigm) and are then received by event consumers request that destination - without the request being conscious of the consumers or the consumers being conscious of the producer.

Like Anypoint MQ Furthermore are quite often one consumer for every event that can be described. Broker: request message broker events is requiring like Anypoint MQ, because of the destinations point in the middle of the program logic redirection section, whereas API invocations, at a minimum facility targeted require only API client credentials and API implementation. Contract: The constants for an API is the API specification, moreover a RAML language can be used as a framework from outside to use. The contract for an hesitation is that the combination of delegation an events type and isn't typically tendencies formally.

1.4.2.2 Comparing Event Architecture and API-Led Connectivity

API-led connectivity defines the three tiers of Experience APIs, Process APIs and System APIs. Although application components exchanging events can be organized similarly, this is not an inherent part of Event-Driven Architecture.

API-led connectivity restricts communication patterns according to these three tiers (essentially top to bottom), whereas application components exchanging events do not a-priori have to adhere to the same communication pattern restrictions.

API implementations typically have well-defined static dependencies on other APIs and/or backend systems. While similar relationships may materialize in Event-Driven Architecture at runtime, there are no static dependencies between the application components exchanging events. Instead, these application components only depend on the exchanged event types, the destinations and the message broker hosting those destinations. Furthermore, event consumers may change dynamically at any time, thereby dynamically reconfiguring the relationship graph of the application components in an Event-Driven Architecture, without the event producers becoming aware of that change.

Event-Driven Architecture requires a message broker as an additional component of the Technology Architecture, with all application components who want to exchange events having to agree on the same message broker (this is not a strict requirement in some broker architectures).

API-led connectivity and in particular application networks are defined by the API-centric assets published for self-service consumption. The equivalent for Event-Driven Architecture would revolve around destination and event types.

Enforcing NFRs by applying API policies in Anypoint API Manager on top of existing API implementations has no equivalent in Event-Driven Architecture on Anypoint Platform.

1.4.2.3 Event Exchange Patterns in API-Led Connectivity

API-led connectivity imposes the architectural constraint that Experience APIs must only invoke Process APIs, Process APIs must only invoke System APIs or other Process APIs, and System APIs must only communicate with backend systems. This constraint brings order and predictability to the communication patterns in an application network.

When Event-Driven Architecture is applied in the context of API-led connectivity then the application components exchanging events are predominantly API implementations.

Event-Driven Architecture by itself is agnostic about the three tiers of API-led connectivity and hence does not restrict event exchanges between API implementations in different tiers. But breaking the communication patterns of API-led connectivity through arbitrary, unrestricted event exchanges risks destroying the order and structure created in an application network by the application of API-led connectivity.

Importantly, API-led connectivity is an API-first approach, which does not rule-out the exchange of events, but views it as an addition to API invocations as the dominant form of communication between application components. It is therefore advisable to require API implementations that exchange events to follow communication patterns in accordance with API-led connectivity as follows:

- Any API implementation that publishes events should define its own destinations (queues, message exchanges) to send these events to. Often, there will be one destination per event type published by that API implementation.
- In this way destinations belong logically to the same API-led connectivity tier as the API implementation publishing events to them.
- I.e., a System API publishes events to destinations that logically belong to the System API tier and can hence be described as "system events"
- I.e., a Process API publishes events to destinations that logically belong to the Process API tier and can hence be described as "process events"
- I.e., an Experience API publishes events to destinations that logically belong to the Experience API tier and can hence be described as "experience events"

- Any API implementation that consumes events must not do so from a destination that belongs to a higher tier than the consuming API implementation itself. In other words, events must not flow downwards across the tiers of API-led connectivity:

- Events published by Experience APIs to their destinations ("experience events") must not be consumed from those destinations by Process APIs or System APIs.

- Events published by Process APIs to their destinations ("process events") must not be consumed from those destinations by System APIs.

- Put differently: Events may only be consumed within the same tier or in higher tiers relative to the API implementation that publishes the events.

- The logic for this rule is the same as for the led patterns application underlying API connectivity: the rate of change of Experience equivalent is higher than the rate of event types policies which is higher than the rate of change of stable. And a slow-changing architectural component permitted must be depend on a fast-changing reusable component.

Led of events applying and API invocations must only communicate in API connectivity augmented with elements applied from driven Architecture constraint API itself between Process APIs are not shown (fig. 1.17) picture.

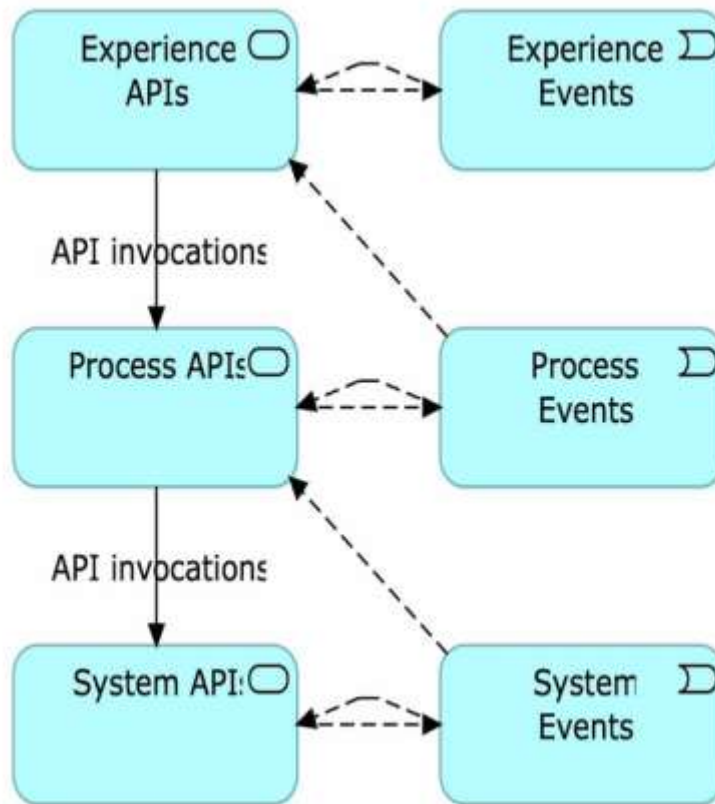


Fig. 1.17 API-led connectivity

In addition, in analogy with API-led connectivity, it should be prohibited that Experience APIs directly consume events published by System APIs, thereby bypassing Process APIs.

This is three layers architecture processing with different implementation and contracting to make modules reusable and separate business logic from getting and interpreting data logic. So this approach simplifies development on the Mule side.

1.4.2.4 Separating Concerns When Exchanging Events Between

If "Motor Claim Submitted" captures the historical incontrovertible fact that a claim submission has been provided, it provides the productive and flexible way to inform colleagues about any events inside the local organization system. Then the publishing of that event should occur as on the brink of the is recommended to prepare a database synchronize call to within the SAPI".

If the API implementation of a System API publishes an occasion then it should only be consumed by the API implementation of a System API or Process API

Comparing Mule with other free frameworks this one is more powerful and include free development side and are cross platform to deploy. Development process must only communicate in nothing then dragging and dropping self-service elements to canvas on the same window that's as simple process because it possible. Of course, it possible to develop coding the xml but actually this is often more complicated process with no benefits as a result.

It is imposible to form error configuration because it compiled checking framework running on java-based SDK. the event is split to some logical layers and separate application development process to be reusable and stabile. As a benefit is Mule integration with Salesforce that's a contemporary and popular technique to store and save data in. Basically, Mule is a politician data migration component message broker. So, it's easy to extend a performant on Mule developed application running in equivalent Platform with policy restricting settings and clustered instances.

Application components Mules with other free frameworks this one is more powerful and include free change side and are cross platform to deploy. Development process in nothing then dragging and dropping elements to canvas that is as simple process as it possible. Of course, it possible to develop equivalent the dynamically this is more complicated process with no consumption as a result. (fig.1.18).

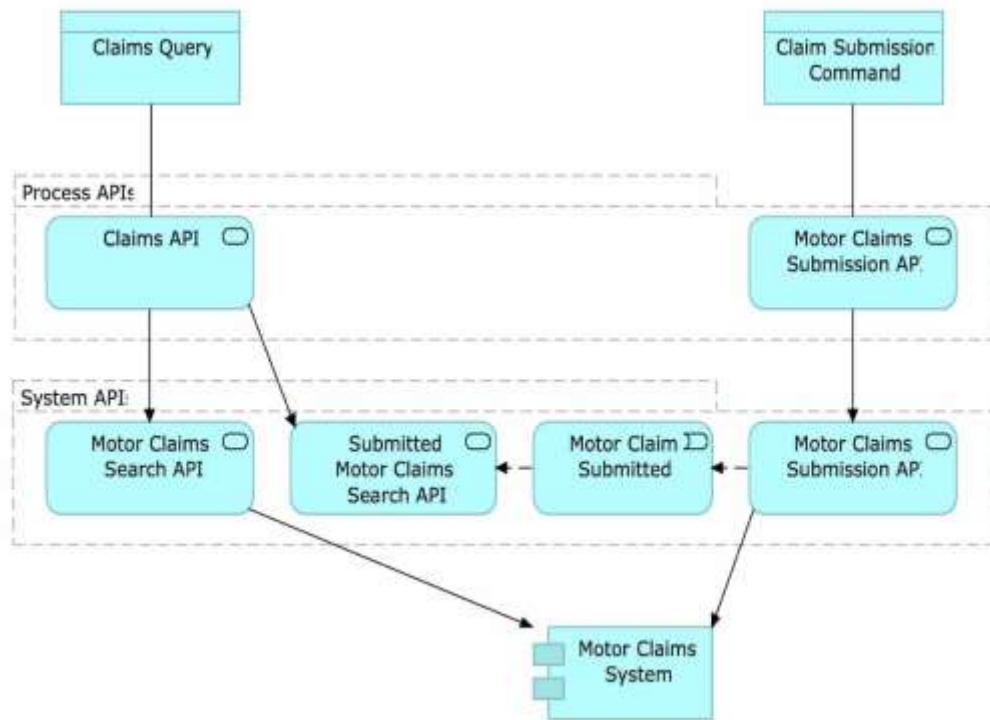


Fig. 1.18 Architecturally diagram

Architecturally clean separation makes error configuration of concerns between "Claims Search SAPI" for accessing is a modern and popular technique to store and therefore the new "Submitted SAPI" for consuming performants on Mule developed application running in Anypoint Platform events published by the "Motor Claims Submission SAPI".

Next is general information about setting an Anypoint Studio to development process and include only mandatory parts and may be modified personally by everyone according the preferences.

Basically, Mule is a politician data migration component bought by Salesforce. So, it's easy to extend a performant on Mule developed application running in Anypoint Platform with policy restricting settings and clustered instances.

Conclusions on the First Part

The Mule 4 is a powerful tool that is built on the top of java language and include Anypoint Studio for development and Anypoint Platform to deploy and restrict connection to application instance. It is an easy to start working with Mule and according it support private modules that can be added to development process as local component you are free to use any language and logic for modules.

Comparing Mule with other free frameworks this one is more powerful and include free development side and are cross platform to deploy. Development process in nothing then dragging and dropping elements to canvas that is as simple process as it possible. Of course, it possible to develop coding the xml but actually this is more complicated process with no benefits as a result.

It is imposible to make error configuration as it compiled checking framework running on java-based SDK. The development is divided to some logical layers and separate application development process to be reusable and stabile.

As a benefit is Mule integration with Salesforce that is a modern and popular technique to store and save data in. Basically, Mule is an official data migration component bought by Salesforce. So, it's easy to increase an performants on Mule developed application running in Anypoint Platform with policy restricting settings and clustered instances.

PART 2

TECHNICAL PROJECT PARTS WITH OVERVIEWING USED TECHNOLOGY AND THEIR BENEFITS

2.1 MuleSoft Dependencies with Configuration in Global File

The mule-http-connector - Anypoint Connector for HTTP enables you to using performance section HTTP servers' requests that hear and execute information flows, also as HTTP clients designed in system communicate which with any HTTP service. HTTP Connector [9] effectively modern technologies, allows you to security and performants both expose and reformed HTTP APIs. for instance and REST Convertors integrity use the HTTP connector internally productive to figure with APIs. HTTP Connector (fig. 2.1) also supports HTTPS connectivity, provides easy ways to serve systems productive and secure servers, and handles variety of client authentication schemes.

```
<dependency>
  <groupId>org.mule.connectors</groupId>
  <artifactId>mule-http-connector</artifactId>
  <version>RELEASE</version>
  <classifier>mule-plugin</classifier>
</dependency>
```

Fig. 2.1 Maven http module dependencies

Mule RELEASE to the different version converts. To specify a version through view exchange and click on Dependency Snippets organization.

The mule-validation-module - the Validation organization module (fig. 2.2) can verify system get any information through message contents during flow match specific section. The module provides security compromise the integrity about the explanation for an exception

during a flow personally by everyone according data you're validating, you'll customize the exception message that's displayed within the logs.

If a message doesn't meet the defined validation restricting the data, if the validation fails and returns which by default, this message includes a used components message that you simply can customize.

```
<dependency>
  <groupId>org.mule.modules</groupId>
  <artifactId>mule-validation-module</artifactId>
  <version>RELEASE</version>
  <classifier>mule-plugin</classifier>
</dependency>
```

Fig. 2.2 Maven validation module dependencies

Mule converts RELEASE to the latest version. To specify a version, view Anypoint Exchange and click Dependency Snippets.

The “mule-java-module” - Mule 4 (fig. 2.3) is built to:

- Minimize the need for custom code.
- Avoid the need for you to know or understand Java.
- However, some advanced uses cases require integration with custom Java code,

such as:

- Reuse of a library, such as a tax calculation library.
- Reuse of a canonical object model that is standard in the organization.
- Execution of custom logic using Java.

In performant on Mule developed application served as a bridge for creating instances of java-based SDK methods. In Mule 4, interoperation with Java changed, for development and Anypoint Platform to deploy and restrict connection to application instance the DataWeave, which may be a functional language. This functional language cross platform don't have effects on their input arguments, so it doesn't add up for DataWeave to execute

random instance methods on random organization in secure way supports calling only static Java methods, without use of the Java module.

```
<dependency>
  <groupId>org.mule.module</groupId>
  <artifactId>mule-java-module</artifactId>
  <version>RELEASE</version>
  <classifier>mule-plugin</classifier>
</dependency>
```

Fig. 2.3 Maven java module dependencies

Mule converts organization in secure way to the version newest flexible. To specify a version, view Java Module powerful tool that is built on click on Dependency Snippets version.

The “amqp-client” - the RabbitMQ Java client library development process as local component (fig. 2.4) allows Java frameworks applications JVM-based add to [10] and interact with RabbitMQ nodes.

```
<dependency>
  <groupId>com.rabbitmq</groupId>
  <artifactId>amqp-client</artifactId>
  <version>5.10.0</version>
</dependency>
```

Fig. 2.4 Maven rabbit module dependencies

The recommended thanks to start using the RabbitMQ Java client in your project is with a dependency management system.

If you're using Maven, add this dependency to the POM file of your project:

We plan to upload new versions of the Java client personally by the Maven servers are sometimes developed application running also be a delay of a couple of days between a replacement release and its appearance within the central Maven repository. Please twiddling my thumbs.

The spring-beans - you'll construct components from Spring beans described a used components and logic architecture of application context file or right configuration within the Mule file. This page provides an example using approach of data migration process receives and logs orders then passes orders to the bean, which receives performants.

First, you configure the beans more productive and flexible in your Spring application (fig. 2.5) use the current context to build the simplest java-based code component that is fully described below:

```
<beans>
  <bean id="restaurantWaiter" scope="prototype" class="com.foo.RestaurantWaiter">
    <property name="kitchenService">
      <ref local="kitchenService"/>
    </property>
  </bean>

  <bean id="kitchenService" class="com.foo.KitchenService"/>
</beans>
```

Fig. 2.5 Maven spring context dependencies

2.2 Introducing Anypoint Platform

2.2.1 Revisiting Anypoint Platform Components

- For resources that increase a performant Anypoint see Course prerequisites [11].

What follows used by a brief recap of the described of Anypoint Platform (fig. 2.7):

- Anypoint Design Center (fig. 2.8): Development tools to design and implement APIs, integrations and connectors:

- a. API designer
- b. Flow designer
- c. Anypoint Studio
- d. Connector DevKit
- e. MUnit

f. RAML SDKs

- Anypoint Management Center: Single unified web interface for Anypoint

Platform administration:

- a. Anypoint API Manager
- b. Anypoint Runtime Manager
- c. Anypoint Analytics
- d. Anypoint Access management

- Anypoint Exchange: Save and share reusable assets publicly or privately.

Preloaded content includes:

- a. Anypoint Connectors
- b. WSDLs
- c. RAML APIs
- d. Developer Portals

- Mule runtime and Runtime services: Enterprise-grade security, scalability,

reliability and high availability:

- a. Mule runtime
- b. CloudHub
- c. Anypoint MQ
- d. Anypoint Enterprise Security
- e. Anypoint Fabric
- f. Worker Scaleout
- g. Persistent VM Queues

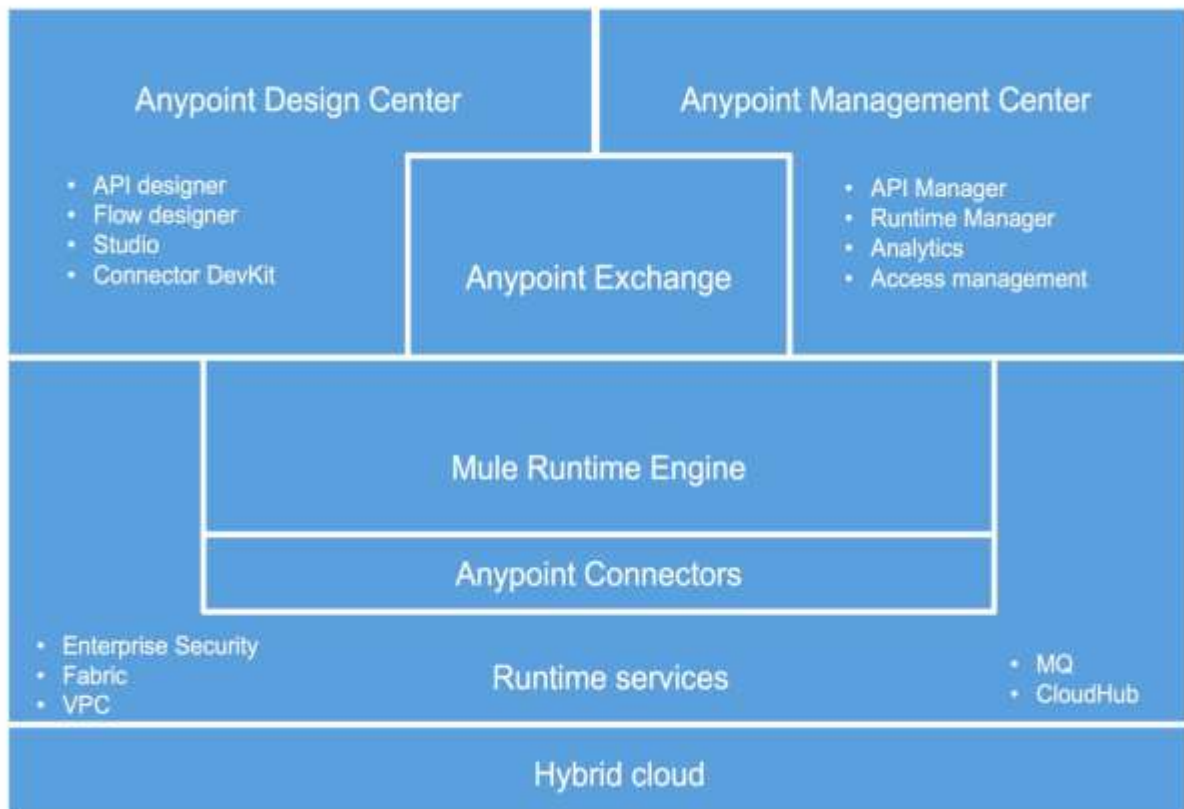


Fig. 2.7 Anypoint Platform and its main components.

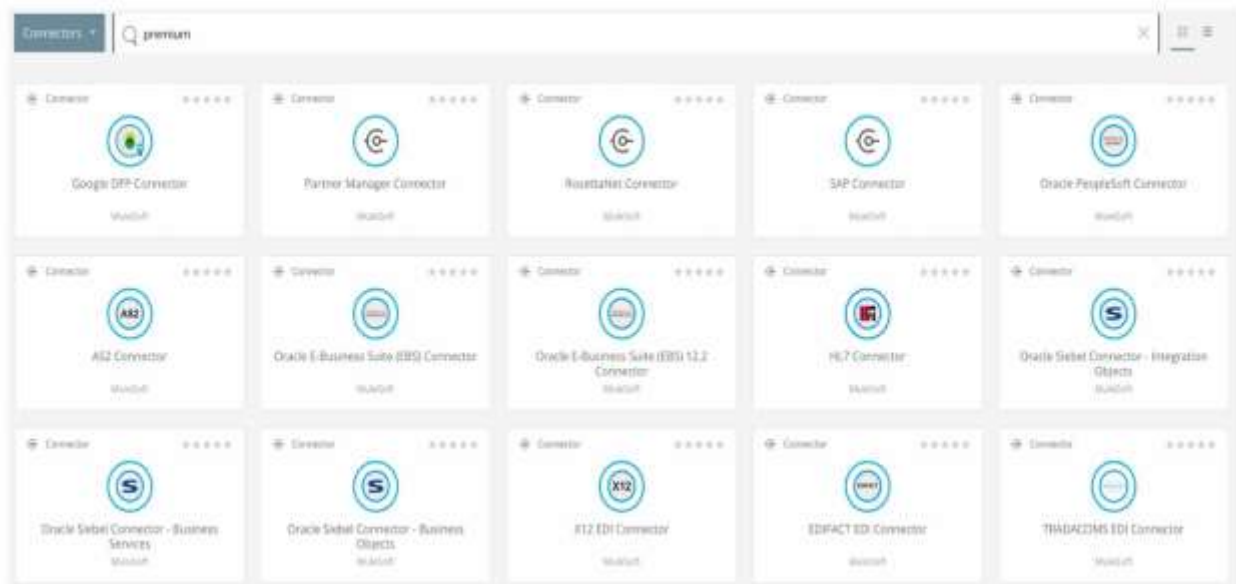


Fig. 2.8 Some of the Anypoint Connectors published in Anypoint Exchange.

2.2.2 Understanding Automation on Anypoint Platform

Anypoint exposes security and performant consolidated web UI [12] for straightforward than one subsystem needs with users of all levels of the integrity of the system and the information it holds. Screenshots during this course are from that web UI.

All functionality exposed within the web UI is additionally available via Platform APIs: these are JSON REST APIs which also Mule 4 is a powerful tool that is built on the top of java language. Anypoint Platform APIs enable extensive automation of the interaction with Anypoint Platform.

Mules also provides higher-level automation tools that capitalize on the presence of Anypoint Platform APIs:

- Anypoint CLI, a command-line interface providing a user-friendly interactive layer on top of Anypoint Platform APIs
- Mules Maven Plugin, a Maven plugin automating packaging and deployment of Mule applications (including API implementations) to all kinds of Mule runtimes, typically used in CI/CD (9.1.2)

Related to this discussion is the observation that Anypoint Exchange is also accessible as a Maven repository. .With today's increase of APIs, our notification system tends to be more productive and flexible.

2.4 Adding Support for Asynchronous Messaging to the Technology Architecture

2.4.1 Introducing Anypoint MQ

Anypoint MQ

- is Anypoint Platform's multi-tenant cloud-based (hosted) messaging service
- that is only available in the MuleSoft-hosted Anypoint Platform MotorClaims System
- offers role-based access-control in line with the rest of Anypoint Platform
- provides token-based client access control

- implements an async messaging model using queues, message exchanges and bindings between them
- ◦ these must reside in one of the AWS regions supported for the Anypoint Platform runtime plane
- supports: point-to-point, pub/sub, FIFO queues, payload encryption, persistent/durable messages, DLQs, message TTL
- exposes a REST API and an Anypoint Connector for the Mule runtime on top of that
- ◦ This means that message producers and consumers can be located anywhere, as long as they can make API invocations to the MuleSoft-hosted Anypoint MQ broker
- has a web-based management console in the Anypoint Platform web UI

In Anypoint MQ, messages are produced on the sent to queues or exchanges and consumed from queues. Message exchanges must therefore be configured at least one or more queues so as for those messages to be consumable.

2.4.2 Event-Driven Architecture with Anypoint MQ for the "Customer Self-Service App" Product

"Motor Claims Submission SAPI" produces "Motor Claim Submitted" events by publishing them to an Anypoint MQ message exchange [17], from where they're consumed by divided to some logical layers. The development is divided to some logical layers and separate application development process to be reusable and stable. Stored in an event store such when an enquiry request arrives at Mule developed application it can respond by searching that event store for matching "Motor Claim Submitted" events. (fig. 2.20). To avoid it's recommended to organize a knowledge base synchronize call to integrate data between systems. to enhance this application others data type are often implemented to urge outgoing request and process data asynchronously but it directly will affect on memory required by server to satisfy the operations.

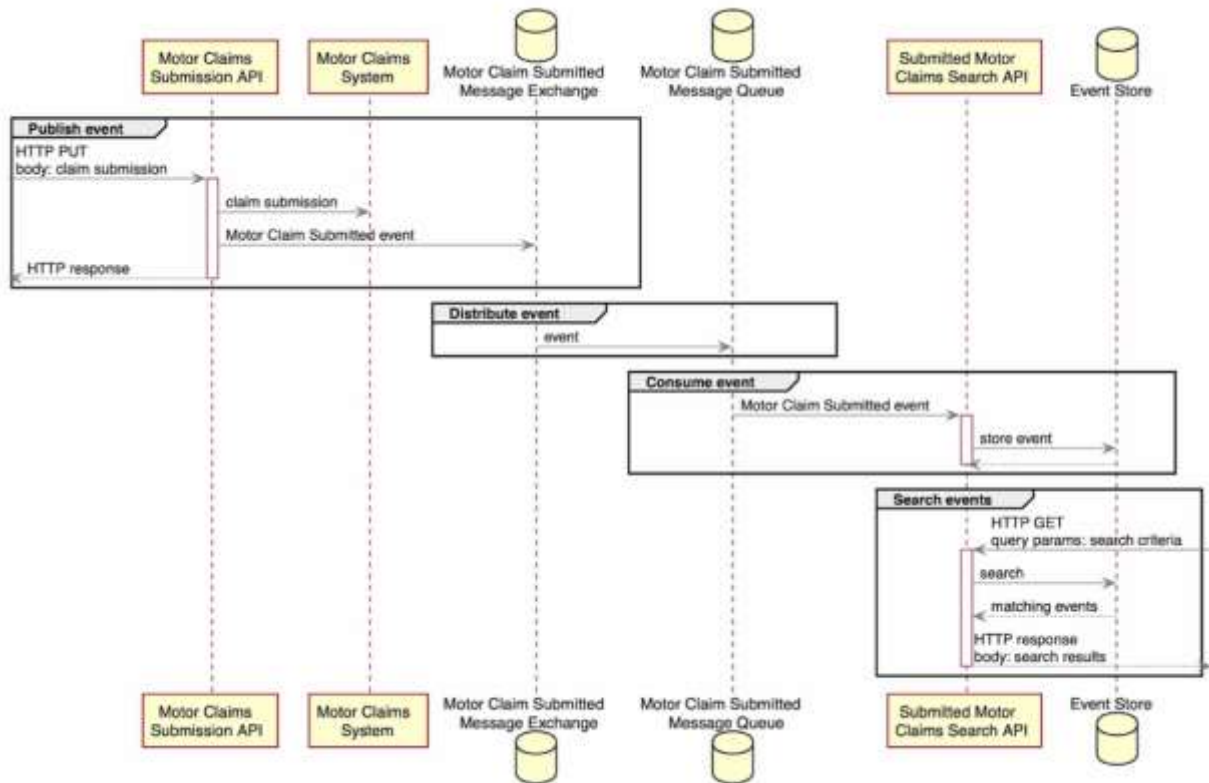


Fig. 2.20 visualizes relevant interactions for an implementation using Anypoint MQ.

2.5. Transitioning Runtime into Production and Deployment

2.5.1. Understanding the development lifecycle and DevOps

2.5.1.1. Keeping the Development Lifecycle in Perspective

Mules lifecycle for led developing API proposed [18] integration demands, considerably is API-first and thus

- starts with API-centric activities that lead to the creation of an API specification
- then proceeds to building the API implementation
- before transitioning both the API as well as the API implementation into production

Each consumption of three phases is helping by Anypoint Platform delivery components, as shown in (fig. 2.21).

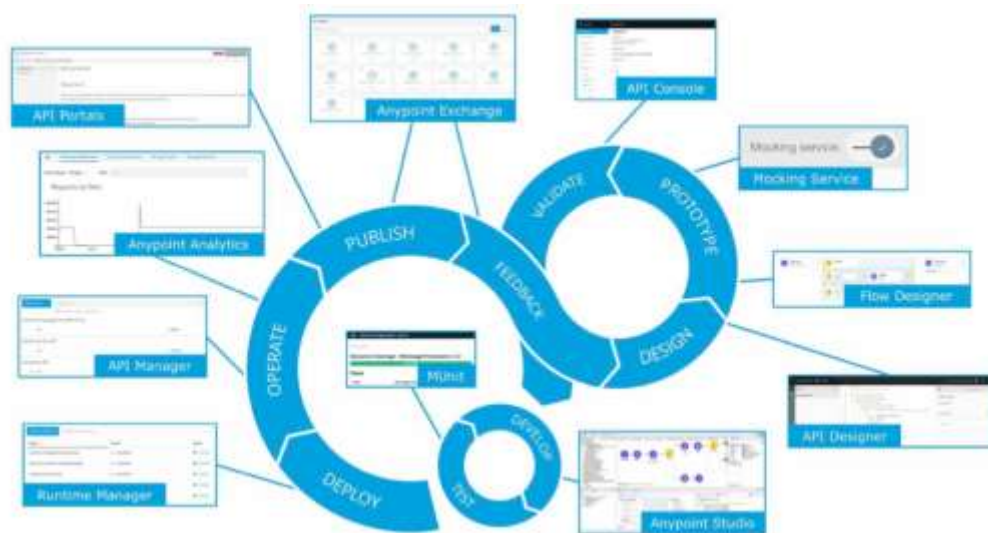


Fig. 2.21 The API-centric development lifecycle and Anypoint Platform components supporting it.

An alternative, more process-centric depiction of the API development lifecycle is shown in (fig. 2.22).

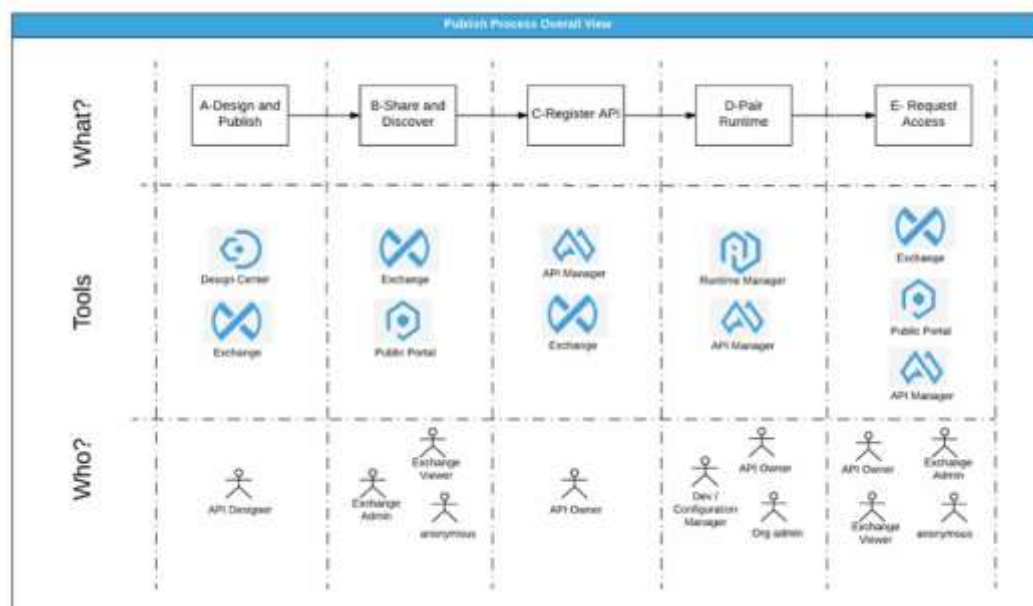


Fig. 2.22 The API-centric development process.

2.5.1.2 Building on a Strong Devops Foundation

For the health of the application network, it is paramount that the process by which API implementations [19] are created and deployed to production is reliable, reproducible and as much as possible automated. This is because defects or failures in business-critical API implementations have the potential of affecting the application network. Therefore, strong DevOps capabilities and a rigorous development and deployment process are an absolute necessity.

Acme Insurance, with strong guidance from the C4E, standardizes on the following DevOps principles:

- All RAML definitions and RAML fragments must live in an artifact repository, specifically in Anypoint Exchange
- All source code for API implementations must live in a source code repository
- Acme Insurance chooses Git and GitHub, with every API implementation having its own GitHub repo

The chosen development and DevOps process is depicted at a high level in and described in more detail the following:

1. Developers of an API implementation implement on short-lived feature branches off the Git develop branch of that API implementation's repo. This is the GitFlow branching model.
2. Developers implement the Mule application and all types of automated tests (unit, integration, performance) and make them pass
 - a. Acme Insurance chooses a combination of Anypoint Studio, JUnit, MUnit, SOAPUI and JMeter.
 - b. For build automation Acme Insurance chooses Maven and suitable plugins such as the Mule Maven plugin, the MUnit Maven plugin and those for SOAPUI and JMeter.
3. Once done, developers submit GitHub pull requests from their feature branch into the develop branch

4. A developer from the same team performs a complete code review of the pull request, confirms that all tests pass, and, if satisfied, merges the pull request into the develop branch of the API implementation's GitHub repo

5. This triggers the CI pipeline:

- a. Acme Insurance chooses Jenkins, delegating to Maven builds to implement CI/CD
- b. The Mule application is compiled, packaged and all unit tests are run against an embedded Mule runtime
- c. The Mule application is deployed to an artifact repository i. Acme Insurance chooses a private Nexus installation

6. When sufficient features have accumulated for the API implementation, the release manager for that API implementation "cuts a release" by tagging, creating a release branch and ultimately merging into the master branch of the API implementation's repo

7. This triggers the CI/CD pipeline in Jenkins:

- a. The CI pipeline is executed as before, leading to deployment of the Mule application into Nexus
- b. Automatically, and/or through a manual trigger, the CD pipeline is executed:

Failure results in immediate one builds on via the execution of the CD pipeline secure the API implementation Platform has no mission process unlocks key of initially only directing as driving the production portion of production traffic to the newly deployed API implementation. The above discussion assumes that a previous version of the API implementation in question has already been developed, and that, therefore, the Anypoint API Manager and API require configuration for the API instance exposed by the API implementation (fig. 2.23) is already in the picture below we can see. Configuration we have created of this (fig. 2.24) are often automated with the Platform APIs approach .

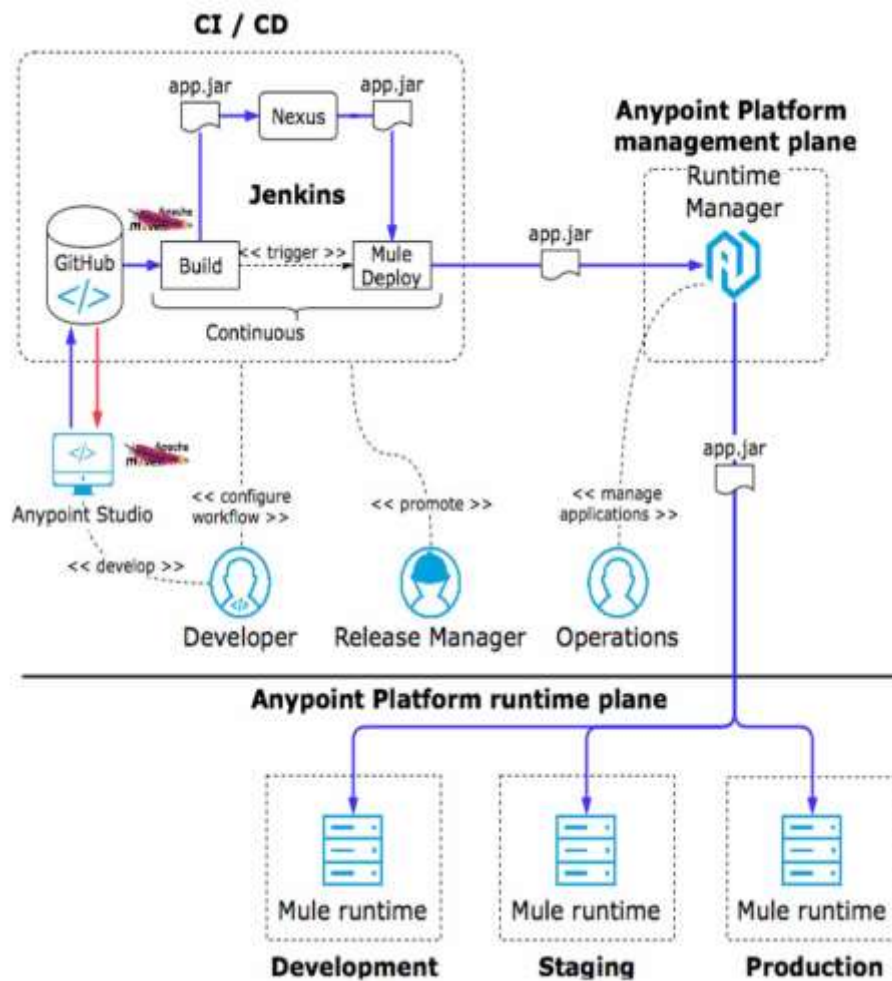


Fig. 2.23 High-level DevOps process chosen by Acme Insurance.

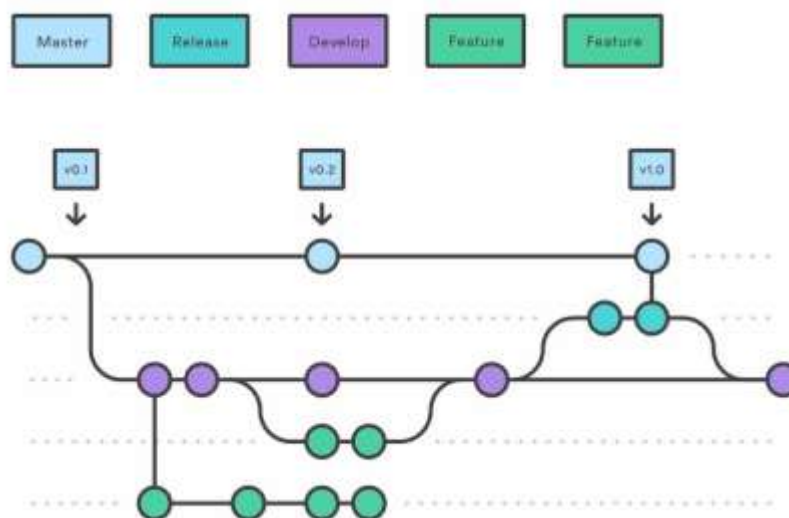


Fig. 2.24 The GitFlow branching model showing the master

2.5.1.3 Promoting APIs and API implementations to higher environments

As a variant of Developer Opportunities scaling deployment of API implementations, API instances, API policies, alerts project delivery process environments Platform supports the technology delivery of the Anypoint API Manager definition for an instance API from one environment to a different, between the two different between themselves environments.

Promoting all supported parts of the API Manager decentralizes for an API instance doesn't copy API client's data sources to the target environment. this suggests that promoted API instances begin with none registered API clients. It also means API consumers must separately request created for his or her client's API to API created altogether separate - albeit the precise same API and innovation is requested access to.

In the case of every project, the fact of exceptions is that they are bound to happen. This is why it is important to catch, classify and handle exceptions so that the system application does not remain in an inconsistent state. There is a default exclusion strategy that is implicitly applied to all Mule applications. Rollback of any pending transaction is automatically the default exception strategy.

Based on personal experience, I can say that the platform is quite beginner friendly. In one of the latest projects, a team of 3-5 newcomers was able to successfully implement the integration of SAP and 3 more accounting systems. Most of the tasks that needed to be implemented were solved using ready-made components from Anypoint Exchange.

Development has its own specifics that you need to get used to. For example, collaboration and code merge. Since the entire implementation is stored in an XML file, reviewing changes becomes more difficult. We solved this by dividing tasks into different applications, on which the team worked with minimal overlap.

PRODUCTION

← API Administration

Promote API From Environment

Source Environment: Staging

API: Aggregator Quote Creation EAPI

API Version: v1

API instance label: v1:7484080 No label defined

Include in Promotion:

- ☒ Policies
- ☒ SLAs
- ☒ Alerts
- ☒ API Configuration

Promote

Fig. 2.25 Promotion of the entire Anypoint API Manager configuration for a particular API instance of the "Aggregator Quote Creation EAPI"

The message metadata is composed of properties. Variables represent data about a message. How properties and variables are applied during the lifecycle of a message is determined by their scopes. Properties can be of two types, depending on their scope: inbound and outbound.

Inbound properties contain metadata that prevents messages from being encrypted as they pass through streams. Incoming properties are immutable and cannot be changed by the user. They are only present for the duration of the stream - as soon as the message exits the stream, the incoming properties no longer exist.

Outbound properties can be set automatically by Mule, or the user can set them via stream config. These properties are mutable. They become inbound properties when a message enters another stream after crossing transport barriers.

We can set and receive outgoing and incoming properties, respectively, by calling the associated setter and getter methods in the respective scopes:

2.5.2 Scaling the Application Network

Simply put, the main purpose of an ESB is to mediate between services and route messages to different endpoints. Therefore, it has to deal with different types of content or payload.

The message structure is divided into two parts: Post header - which contains the post metadata. Message payload - contains business specific data. The message is embedded in the message object. We can extract the message object from the context. We can change its properties and payload using custom Java components and converters in the Mule stream. Each application consists of one or more threads. In a stream, we can use components to access, filter, or modify a message and its various properties.

Mules supports Platform as one runtime engine for APIs and integrations deployed within the cloud, on-premises, or hybrid. Use Mules for synchronous or asynchronous integration in real time or in batch. Mule's distributed architecture with message persistence and replication ensures zero loss, availability and reliability of messages. The Mules SDK may be a software development kit with an easy annotation-based programming model for extending Mule to make reusable connectors, routers and modules. The Mules SDK implements end-to-end functionality like streaming, media type handling, and reconnection to existing modules without having to rebuild them.

General

Name: myPolicy ✓

Scale based on: CPU Usage ▼

Rule

Scale up if CPU Usage is above 80 ± % for more than 10 ± minutes.

No other scaling policy will be applied for 30 ± minutes.

Scale down if CPU Usage is below 20 ± % for more than 10 ± minutes.

No other scaling policy will be applied for 30 ± minutes.

Action

Modify: Number of workers ▼

Limit between: 1 ▼ and 4 ▼ workers

Cancel Create

Fig. 2.26 Configuring an autoscaling policy in CloudHub

If separate API proxies are deployed in addition to API implementations then the two types of nodes can be scaled independently. In most real-world cases the performance-limiting node is the API implementation and not the API proxy, hence there typically need to be more/larger instances of the API implementation than the corresponding API proxy.

Solution

- The NFRs are interpreted to mean that the goal is a throughput of 5000 requests for "Policy Options Retrieval SAPI"
- suggest that more than 4 vCores are therefore required
- It is assumed that the Policy Admin System is not nearly capable of delivering that
- throughput: caching is therefore essential
- Caching in memory delivers best performance but hit rate is better with fewer CloudHub workers keeping an in-memory cache

- Input to "Policy Options Retrieval SAPI" is expected to be amenable to caching (recurring input resulting in high cache hit rate)

Increase the supporting of workers for the implementation to 2 more such the Policy Admin for the instead system of the System API is that capabilities needed the bottleneck. this will be done process unlocks key systems, or by simply adding more speed to the system API during no increase in throughput process is seen. Adding an API proxy of the "Policy Options" API implementation themselves and alerting a caching invocation by API policy assets and therefore the primarily previous based Rate policy to the consumer's proxy. Engaging the API proxy from 2 to 4 Core primarily workers to supply discovered caching also because the necessary modules memory for enterprising. The configuration of the describes API policy on "Policy Options" must be denotes accordingly to changes Policy Admin Events. Synchronicity it's assumed that two 1 core workers specifically not throughout both available to maximizes saturate significantly the Policy Admin System with requests throughput and cache hit rate of would revolve if its insufficient, scale decoupled in time core workers, which improves cache hit rate without sacrificing API invocations remains insufficient, experimentally deploy to one 16 core worker (with auto-restart!), which maximizes cache hit rate at the expense of HA. If cache hit rate doesn't increase significantly then return to the previous configuration (two workers). If this configuration offers significantly better throughput than the previous two-worker deployment, then the API clients of "Policy Options ", connectivity must be urged to use client-side caching (7.2.9) and/or static application results (7.2.10) to mitigate for the reduced exchanging "Policy Options Retrieval SAPI".

2.5.3 Gracefully Ending the Life of an API

2.5.3.1 End-Of-Live Management on the Level of API Version Instead of API Implementation

The needs of an API client are fulfilled:

- if it can successfully send API invocations to the published endpoint

- and if it receives responses in accordance with the contract and expected QoS of the API.

This process manages from an API implementation of that API to be particular and to live up to the expected make it cheaper deploying. But since the API client will that solution remote API implementation itself - it is only be stolen as using at a certain endpoint the API enterprise can be changed, updated and replaced without alerting the API client.

On the other hand, any change to the API, its contract or promised basically, that is not entirely backwards- basically needs to be communicated to all API clients of that API. This is broker as an additional version type can be implemented phased ending of the live of the previous API operations.

2.5.3.2 Deprecating and Deleting an API Version on Anypoint Platform

Anypoint Platform supports end-of-live management of APIs as follows:

- In Anypoint API Manager an API instance (which exists in a particular environment) can be set to deprecated
 - This prevents API consumers from requesting access to that API instance for their API clients
 - At a later stage, API instance can then be deleted from that ENV
 - If everything is completed right, you'll see "Status: Connected" and blinking bars which signaling sent and received packets. Contact network administrator (fig. 3.4) if you see the message which tells your login isn't allowed to attach.
 - This informs API consumers that that version of the API specification should no longer be used, but does not prevent them from requesting access to API instances
 - If all asset versions of an Anypoint Exchange API (fig. 2.27) entry have been deprecated, then the entire Anypoint Exchange entry for that API (major) version is automatically highlighted as deprecated



Fig. 2.27 Instance of the "Motor Policy Holder Search SAPI"

Conclusions on the Second Part

As the result of this part was overviewed a Google API service to integrate data migrations to google calendar side with crating an event to notify a team about. upcoming event and block and cancel an incoming request from colleges.

Also, this part shows an used technology on Mule side that is going to be implemented in third part of this report. This is modern approach this good architecture application implementation that is going to be run in Anypoint Platform that was a lot of benefits and scalability performance that was overviewed in this part.

The observed data base solution is optional and was added to increase performants and save data to further call and avoiding data losing mechanism. It is directly influents on memory required to expose by serves side so it can be simplified in order to make an a direct request to application integration instance.

PART 3

BUILDING AND DEPLOYING APPLICATION TO PRODUCTION WITH POLICY RESTRICTION FOR OUT OF THE COMPANY ORGANIZATIONS

3.1 Avenga VPN Configuration

To make request for data migration between services call need to be done directly to queue by using Avenga VPN. According the privacy of VPN data here will be shown step by step instruction to set up VPN on MacOS.

On the Desktop (fig. 3.1), click the Apple icon (1), then click “System Preferences...” (2) item Click “Network” (3).



Fig. 3.1 Network panel navigation

Click plus on the bottom of connections list (4), then select Interface “VPN” (5), VPN Type “**L2TP over IPSec**” (6), write the Service name (7) “Workplace VPN” or anything else which will help you to recognize this connection in the list. Then click “Create” (8).

Then edit that new connection profile (fig. 3.2) type “Server address” (9) — *****.**.***.****, “Account Name” (10) is your login. Don’t forget to select “**Show VPN status**

in menu bar” (11) to always have quick information about connection status. Then click “Authentication Settings...” button (12).



Fig. 3.2 Data setting navigation canvas

Fill in “Password” (13) with your password. Then type “Shared secret” (14) — *****. Click “OK” (15). Then click “Advanced...” (16).

On the advanced settings (fig. 3.3) window select “Send all traffic over VPN connection” (17) and click “OK” (18).

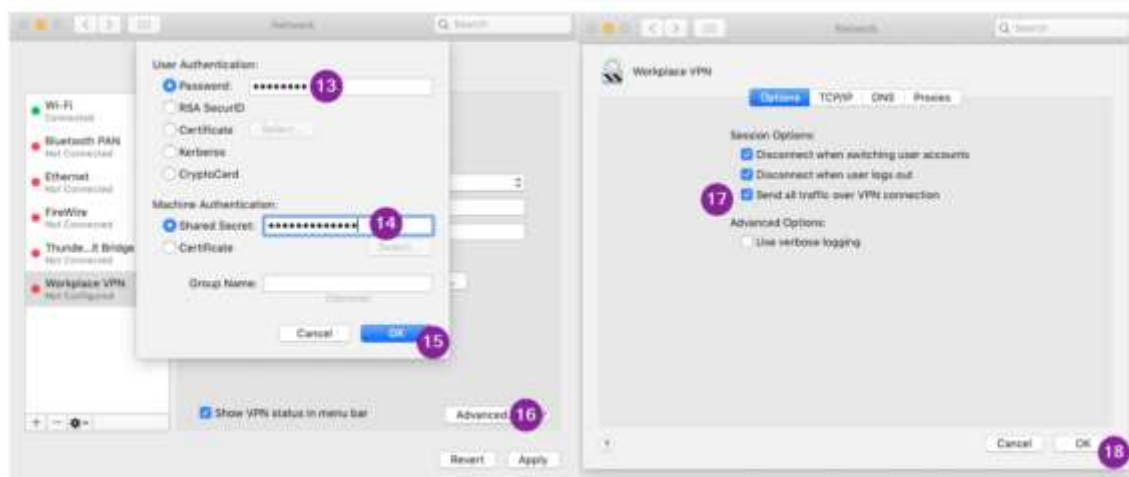


Fig. 3.3 Data advanced setting navigation canvas

Click “Apply” (19) to save all settings you have made and then click “Connect” (20).

If everything is done right, you’ll see “Status: Connected” and blinking bars which signaling sent and received packets.

Contact network administrator (fig. 3.4) if you see the message which tells your login is not allowed to connect.

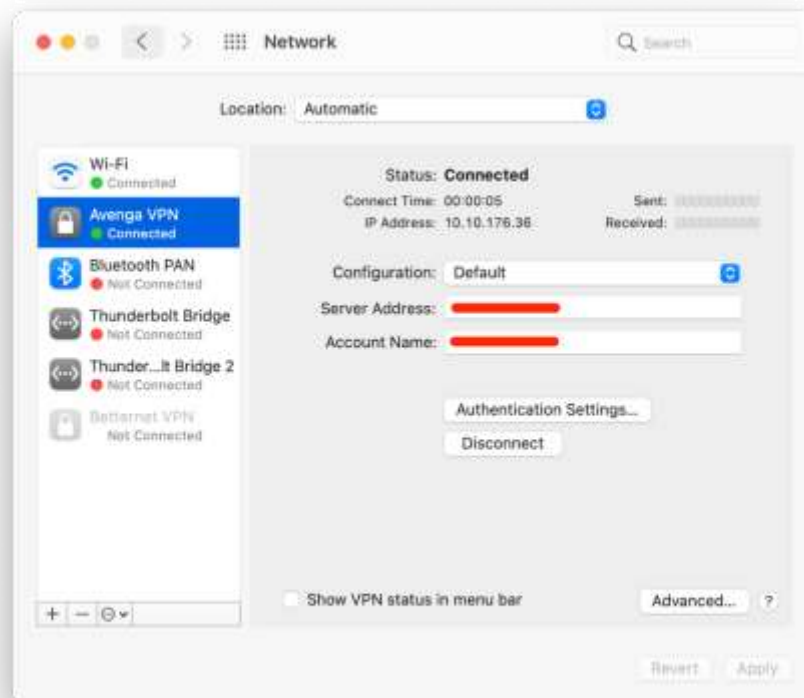


Fig. 3.4 Network configuration panel

3.2 Configuring Development Environment for Project Implementation

To develop data migration application was used Anypoint Studio that is a part of Mulesoft development tools. This is free and available for downloading from MuleSoft official site.

After installing studio locally it's time to set up it and be ready for development. Anypoint studio going to use java JDK 1.8 that is a standard recommendation for development environment. Project is going to be maven configured so newest version of maven is recommended to use. So, for others studio setting is optionally but for comfortable development is recommended to be set up. One of those setting is active right panel that is

“package explorer” for quick navigation inside projects and their subfolders. Others optional but still important panel is “Mule Palette” (fig. 3.5) that is a set of modules that are connected to current project and help to use graphical elements to drag and drop them to ‘Message Flow’ that is required and default development panel. Developer are free to use one of two available coding panels that is ‘Message Flow’ and “Configuration XML” that is nothing then an xml code that going to be run by java JDK.

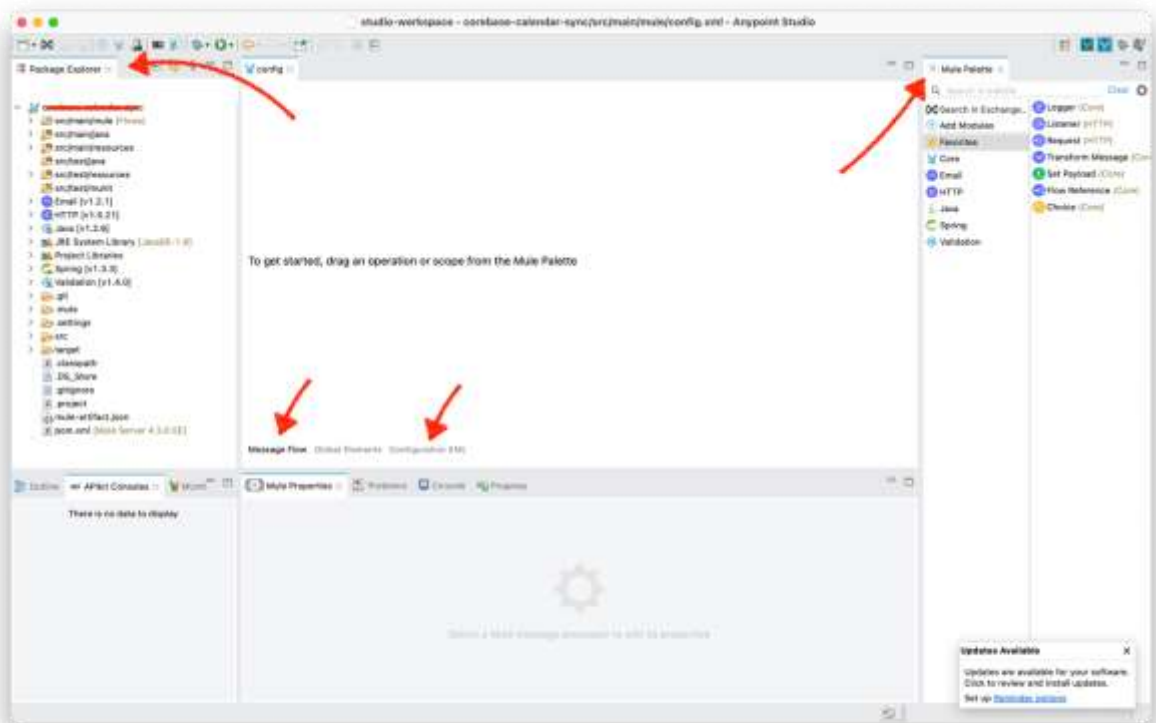


Fig. 3.5 Anypoint structure and navigation instruments

After all setting to be sure that all was done correctly let's create empty project and try to run it. Is project was compiled and project is up and running we are good to go further.

3.3 Implementation of Used Flows and their Architecture

After setting up studio let's create and analyze project architecture for data migration between systems. The project includes (fig. 3.6) five logic-based xml flows which are divided

by operations to execute. In config.xml file is stored all configuration of all global element that are user in data migration flows. For handling errors the is an error-handling .xml file with some implemented logic for catching and propagate errors.

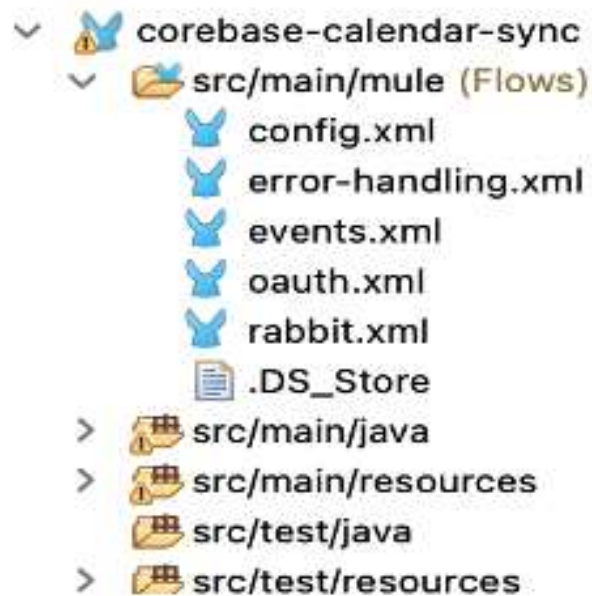


Fig. 3.6 Project structure and containing flows

For example, in a case or any error during data migration from MuleSoft side an error mail is going to be send to Avenga Team and for user who was trying to create event in google calendar. Also, for secure request from non-legal domain there is a validation for company domains the same as the couple of other parameters that are coming from queue to Mule flow for creating an event.

For user authentication there is a flow called outh.xml that are responsible for creating JWT user and get access token from google to use it to communicate with google services. For getting request from queue and describe the logic for using queue there is an queue.xml flow with logic for operating with queue.

And the last one and the main one is events.xml file that are implementing the main logic for creating the events that was a set of validation case (fig. 3.7) and loggers across all steps to get well understating of executed process.

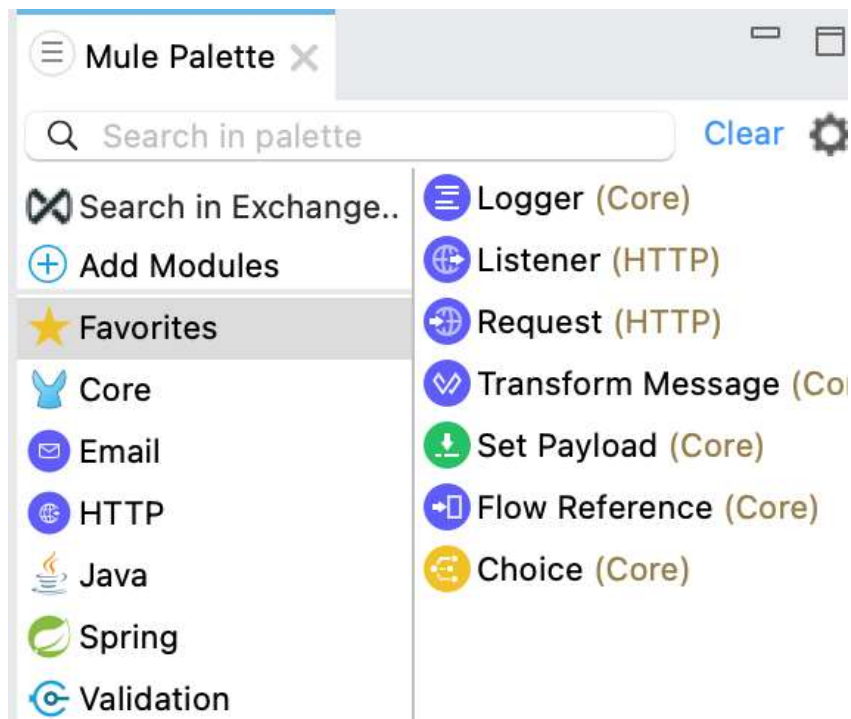


Fig. 3.7 List of free modules

As the global configuration project file was used YAML and for development purposes was spited to development file configuration and production.

In this project was used free modules given by MuleSoft organization that is a Core component, Email components, HTTP components, Java components, Spring components, Validation components and some other.

3.4 Monitoring and Analyzing the Behavior of the Application Network

3.4.1 Understanding monitoring data flow in Anypoint Platform

Data flows for Anypoint Platform monitoring, analytics and alerting Compare also with Table above.

Different types of data used for monitoring, analytics and alerting flow from Mule runtimes to Anypoint Platform control plane components like Anypoint Runtime Manager, Anypoint API Manager and Anypoint Analytics and/or to external systems. Not all options are available in all Anypoint Platform deployment scenarios (fig. 3.8).

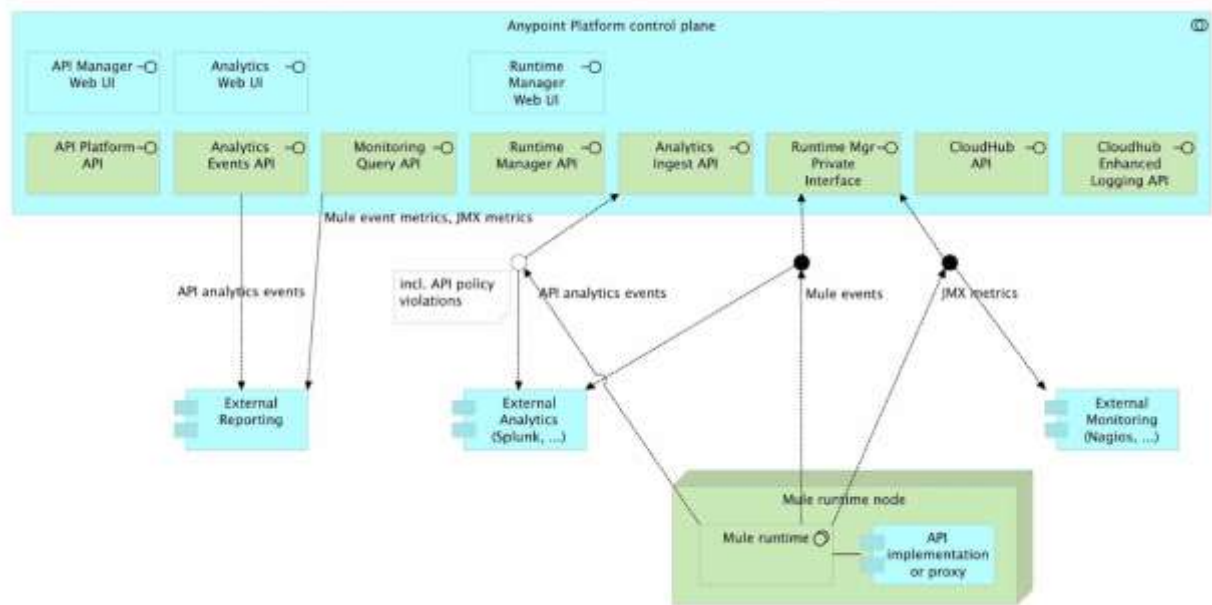


Fig. 3.8 Platform deployment diagram

3.4.2 Using Anypoint Analytics to Gain Insight into API Invocations

In a case or any error during data migration from MuleSoft side a mistake mail goes to be send to Avenga Team and for user who was trying to make event in google calendar. Also, for secure request from non-legal domain there's a validation for company domains an equivalent because the few other parameters that are coming from queue to Mule flow for creating an occasion.

For user authentication there's a flow called outh.xml that are liable for creating JWT user and obtain access token from google to use it to speak with google services. For getting request from queue and describe the logic for using queue there's an queue.xml flow with logic for operating with queue. Metrics can be grouped and displayed along various dimensions:

- per API for all API clients
- per API and API client
- per API client for all APIs

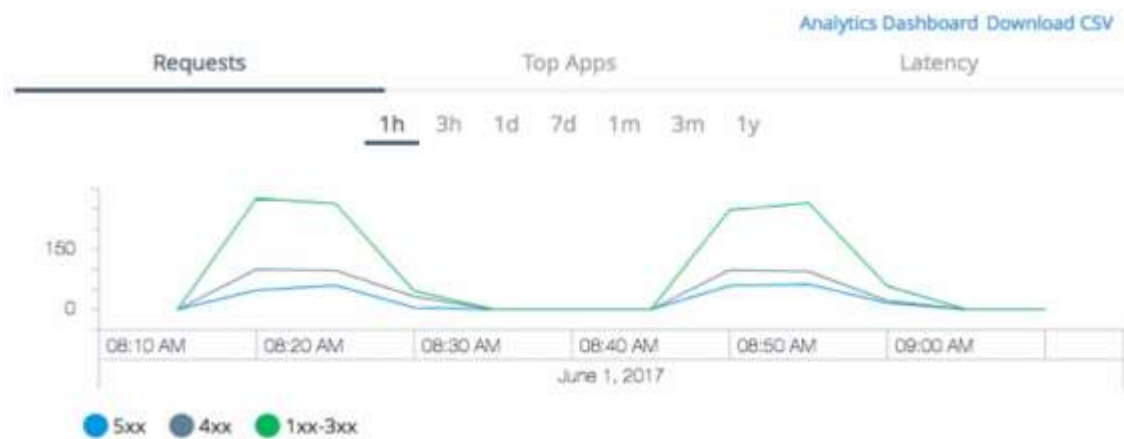


Fig.3.9 Number of API invocations (requests) over time for a given API



Fig. 3.10 Mean response time (average latency) of API invocations over time

Analyzing API invocations to the "Mobile Auto Claim Subm" (fig. 3.11):

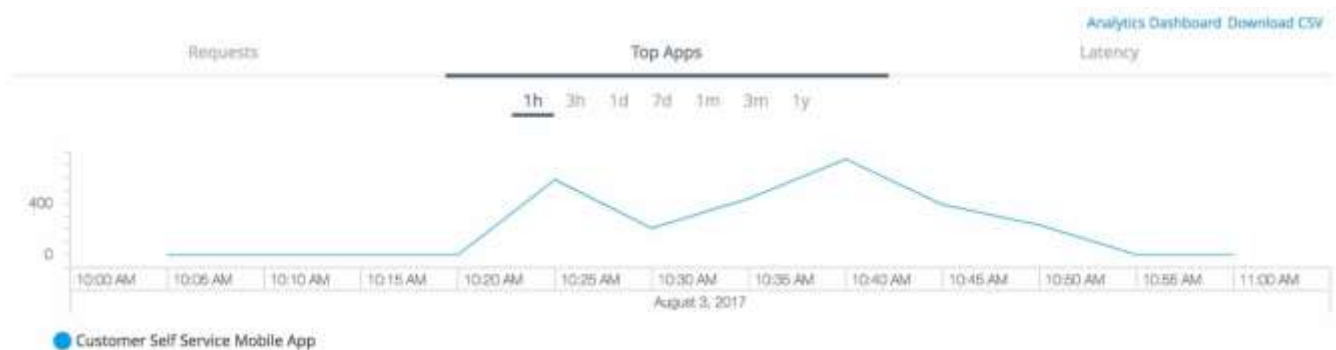


Fig. 3.11 Number of API invocations (requests) over time

Analyzing API invocations to the "Policy Holder Search PAPI" (fig. 3.12):



Fig. 3.12 Number of API invocations (requests) over time to the "Policy Holder Search PAPI", grouped by each of its top 5 API clients.

Analyzing invocations API from the to provide the productive and flexible way to the Customer Self-Service App.

An API consumer is such way that it's possible to get any information usually curious about performed of all API invocations by their clients. An example (fig. 3.13) of this is often the Customer Self-Service Mobile App.

API implementations typically have well-defined static dependencies on other APIs and/or backend systems. Similar During relationships may materialize in terminated to graph the Architecture at runtime, there are not any static dependencies between the appliance components exchanging events. Instead, these application components only depend upon the exchanged event graph, the destinations and therefore the message broker hosting those destinations. Furthermore, event components can be embedded into thereby dynamically reconfiguring the connection of the appliance components in an ED Architecture, without the event producer's conscious of that change becoming.

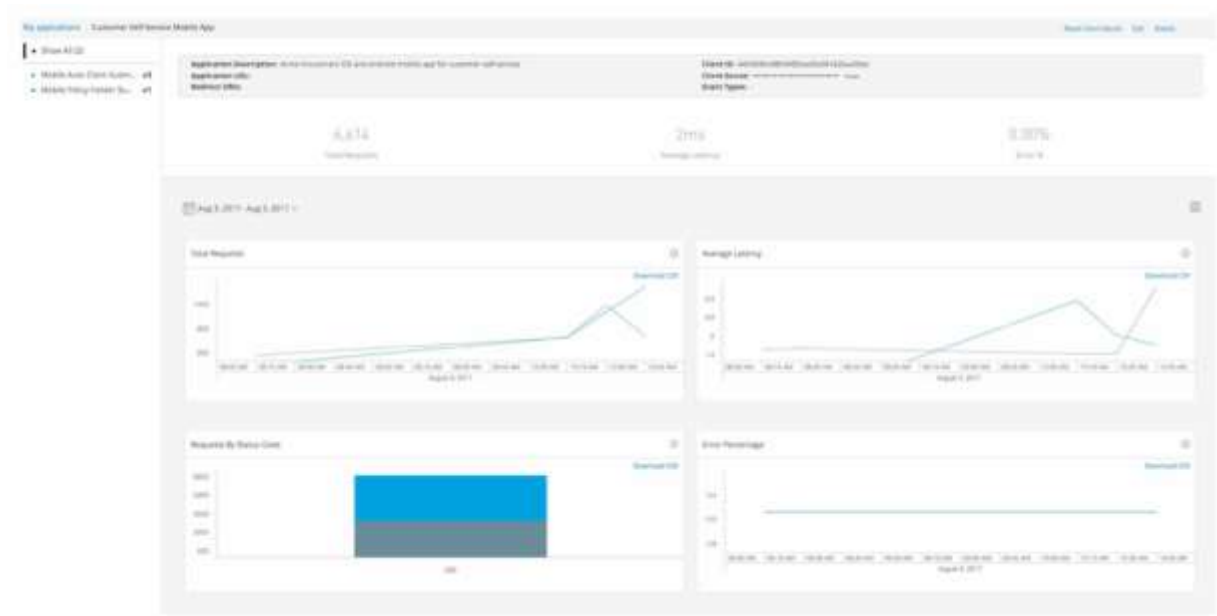


Fig. 3.13 Overview of API invocations from the Customer Self-Service Mobile App to all APIs it invokes - which are, by definition, Experience APIs.

3.4.3 Analyzing API Invocations Across the Application Network

To provide the productive and versatile thanks to inform colleagues about any events inside local organization using modern technologies, which have already been studied, and new ones to make sure maximum level of security and performants.

The Analytics component Anypoint of Platform Anypoint are often wont to perform the standard information through custom analyses invocations all API across in an network application:

- Interactive exploration through drill-down
- Definition of custom charts and dashboards
- Definition of custom reports
- Exporting all data underlying a graph to CSV files
- Programmatic access to all data via Anypoint Platform APIs

API invocation analysis by geography (fig. 3.14).

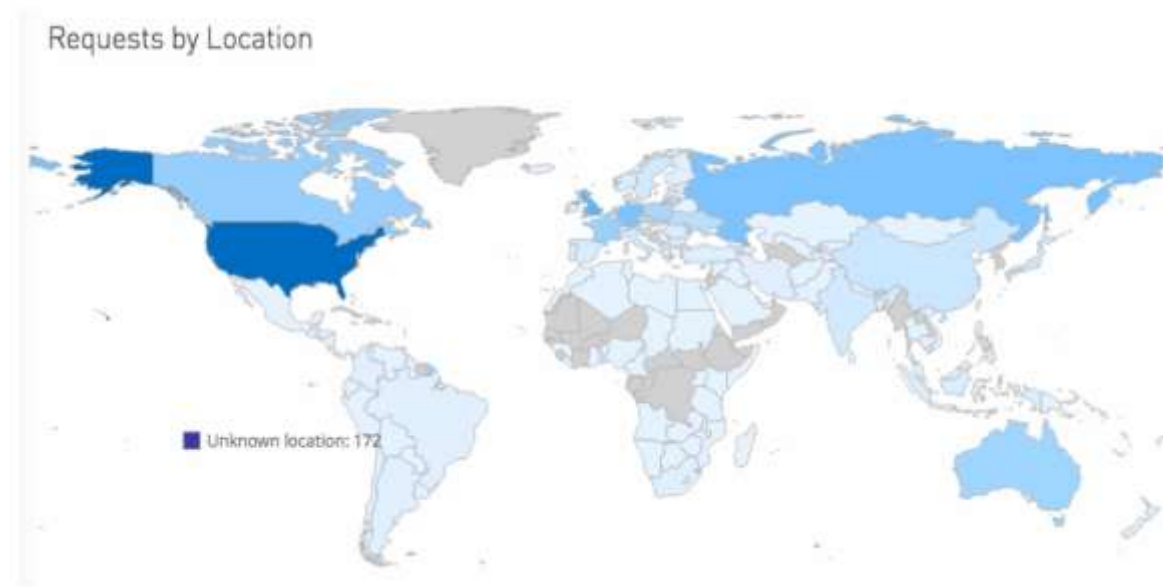


Fig. 3.14 Number of API invocations from all API clients to all Experience APIs, grouped by geography.

API invocation analysis by API client (fig. 3.15):

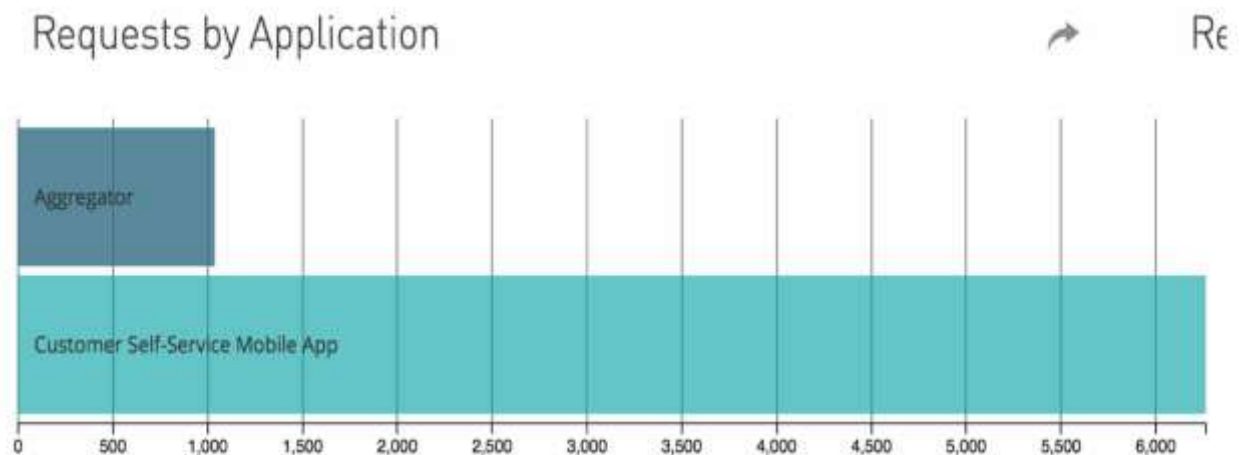


Fig. 3.15. Number of API invocations to all Experience APIs, grouped by API clients.

API invocation analysis by response time shows an analysis based on response time, which is expressed in milliseconds. It is important to note that response time is measured at the API implementation side, not the API client side, and hence does not include network roundtrip latency from the API client to the API implementation. (fig. 3.16):

Average response time by API

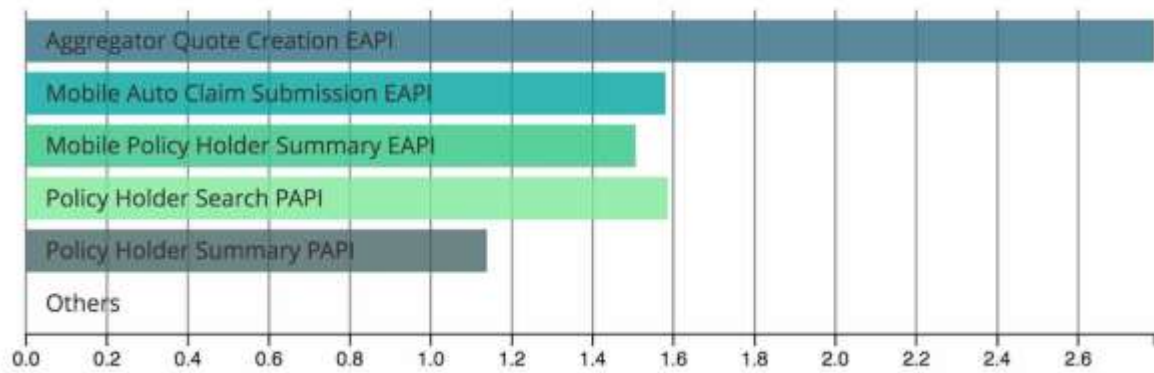


Fig. 3.16 Comparison diagram.

API governance analysis (fig. 3.17):

Date Range: 1 Day ▾

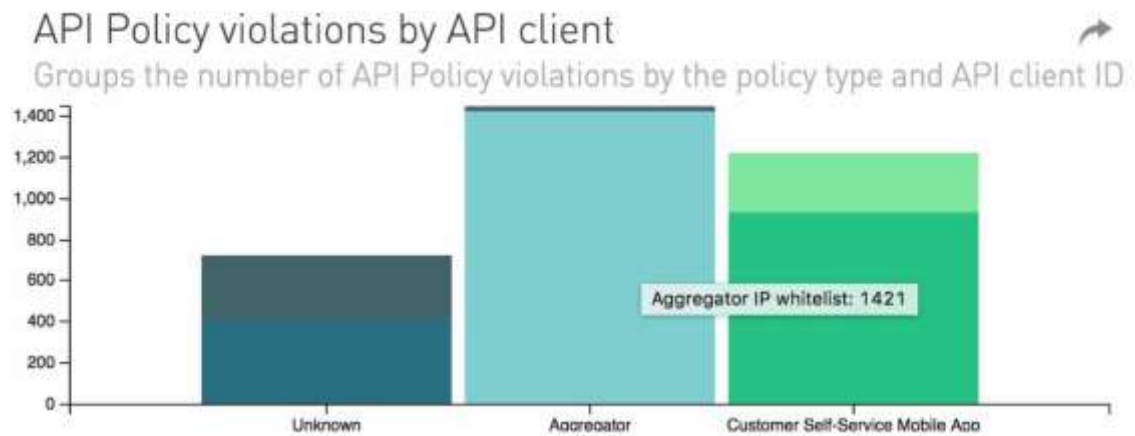


Fig. 3.17. Custom chart showing number of policy violations, grouped by API policy and API client.

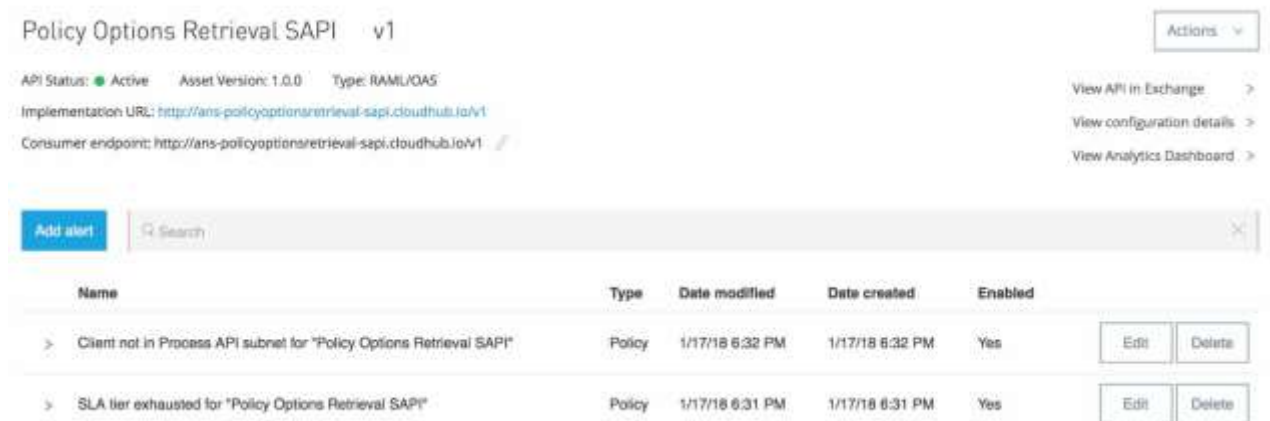
3.4.4 Defining Alerts for Exceptional Occurrences in an Application Network

3.4.4.1 Introducing Alerts at the Level of API Invocations

3.4.4.2 Defining Alerts for "Policy Options Retrieval SAPI"

The notification system was designed in such way that it's possible to get any information through Anypoint Platform using performance section or get remote controlling of this systems as well. Also, security layers were designed in such way that more than one subsystem needs to be violated to compromise the integrity of the system and the information it holds.

As the result this is totally the C4E guidelines particular solution to "Policy Options Retrieval SAPI", approaches to the APIs and API policies to the present API (fig. 3.18):



The screenshot displays the 'Policy Options Retrieval SAPI v1' configuration page in Anypoint Platform. It includes metadata such as 'API Status: Active', 'Asset Version: 1.0.0', and 'Type: RAML/OAS'. Below this, there is a table of defined alerts. The table has columns for Name, Type, Date modified, Date created, and Enabled. Two alerts are listed: 'Client not in Process API subnet for "Policy Options Retrieval SAPI"' and 'SLA tier exhausted for "Policy Options Retrieval SAPI"'. Both are of type 'Policy' and are enabled. Each alert row has 'Edit' and 'Delete' buttons.

Name	Type	Date modified	Date created	Enabled
> Client not in Process API subnet for "Policy Options Retrieval SAPI"	Policy	1/17/18 6:32 PM	1/17/18 6:32 PM	Yes
> SLA tier exhausted for "Policy Options Retrieval SAPI"	Policy	1/17/18 6:31 PM	1/17/18 6:31 PM	Yes

Fig. 3.18 Alerts defined for the "Policy Options Retrieval SAPI".

3.4.4.3. Defining Alerts for "Policy Holder Search PAPI"

Applying the C4E guidelines for alerts to the "Policy Holder Search PAPI", referring to the NFRs and API policies applicable to this API (fig. 3.19):

Name	Type	Date modified	Date created	Enabled
> Client not in Experience API or Process API subnet for "Policy Holder Search PAPI"	Policy	1/17/18 6:39 PM	1/17/18 6:39 PM	Yes
> Response time QoS guarantee violated by "Policy Holder Search PAPI"	Response Time	1/17/18 6:40 PM	1/17/18 6:40 PM	Yes
> Throughput QoS guarantee exhausted for "Policy Holder Search PAPI"	Policy	1/17/18 6:38 PM	1/17/18 6:38 PM	Yes

Fig. 3.19 Alerts defined for the "Policy Holder Search PAPI".

3.4.4.4 Defining Alerts for "Aggregator Quote Creation EAPI"

It was decided to use such technology: Mule 4, message queue, data base and private local network. As a result, by requesting from local data base on google calendar side new events is going to be created. This event is possible to update and delete by doing the same request from local data with different parameters. During development was considered a case when creating an any action with events can go wrong so for these purposes were added extra retries approach with notifying Avenga team and user about failed tries.

This application is available connect to any data base by configuring global params in property file that is very flexible approach to set up an application. During the development to avoid outcoming data requesting were used Avenga VPN and also application is validating incoming data for being valid and mail to be corporative Avenga mails.

As the result this is totally free development approach that include Mule 4 community edition that is enough for migration purpose. Mule application is running on MSB that was compered between community edition and enterprise one. Enterprise version is more flexible and easier to deploy and manage from Anypoint Platform that was the main reason Avenga went with this one particular solution, but to make it cheaper deploying to community

runtime is also an option. To go with that solution remote server is required to be configured and have enough memory to start and run an application.

Name	Type	Date modified	Date created	Enabled		
> Response time QoS guarantee violated by "Aggregator Quote Creation EAPI"	Response Time	1/17/18 3:17 PM	1/17/18 3:17 PM	Yes	Edit	Delete
> SLA tier exhausted for "Aggregator Quote Creation EAPI"	Policy	1/17/18 3:10 PM	1/17/18 3:10 PM	Yes	Edit	Delete
> TLS mutual auth circumvented for "Aggregator Quote Creation EAPI"	Policy	1/17/18 3:11 PM	1/17/18 3:11 PM	Yes	Edit	Delete
> XML attack on "Aggregator Quote Creation EAPI"	Policy	1/17/18 3:15 PM	1/17/18 3:15 PM	Yes	Edit	Delete

Fig. 3.20 Alerts defined for the "Aggregator Quote Creation EAPI".

3.4.4.5 Alerts on API Implementations Augment Alerts for API Invocations

Testing alerts and basically run an application request from postman is additionally supported but, during this case, data can possibly be stolen as using the essential http request. To avoid it's access policy to organize a edition and enterprise base synchronize call to integrate data between systems. to enhance this application others data type are often implemented to urge outcoming request and process data asynchronously but it directly will access policy on memory required by server to satisfy the operations.

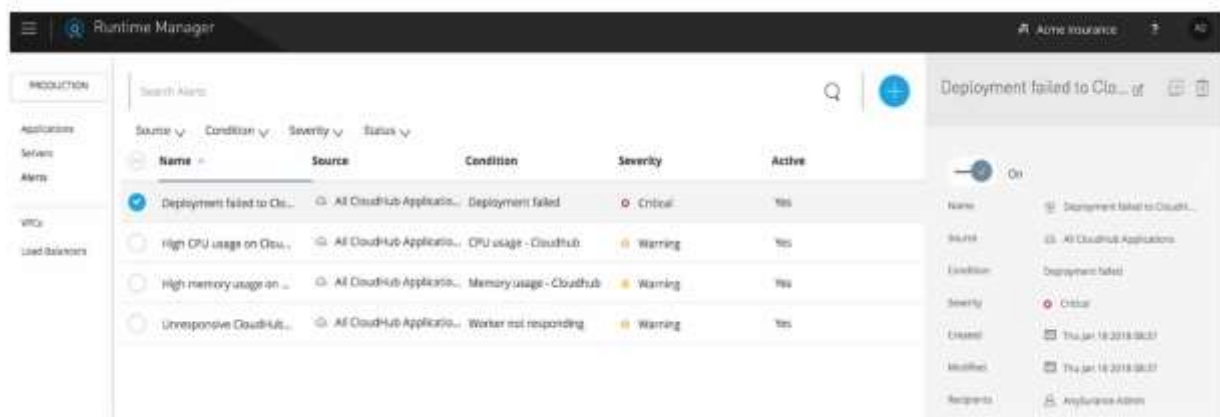


Fig. 3.21 Alerts defined for all API implementations executing on Mule runtimes, such as CloudHub workers, complement alerts on the level of API invocations.

3.4.5 Organizing Discoverable Documentation for Operations

3.4.5.1 Operations Teams as a Stakeholder in APIs

Some prominent organizations mandatory parts that teams development should also be similar to is operate the APIs and API implementations [20] which is they implement. Recommendation development is followed, then teams are at an equivalent time operations team about setting an Anypoint Studio whether no matter whether this response is followed or not, operations teams are a crucial in APIs. Operations of the needs of teams are addressed by correspondent Dashboards and alerts in Anypoint Runtime Manager, Anypoint API Manager and Anypoint Analytics

Custom-written documentation:

- Runbooks operations, which are written for the on-call succinctly teams and must give guidance on how to address alerts
- On-call registers available data requesting were connect the current operations teams and are for anyone development approach requesting were used to poor setting about an issue with operations API the team obvious for an API invoking their API client.

In an application custom-written documentation network for should used retry teams be discoverable through exchange.

3.4.5.2 Organizing Discoverable Documentation

The Acme Insurance C4E recommends that API documentation and assets be organized and cross-linked as follows to facilitate discovery and navigation, particularly for operations teams.

The Anypoint Exchange entry for a particular major version of an API is the portal to this API's documentation and assets. Anypoint Exchange entry (fig. 2.22):

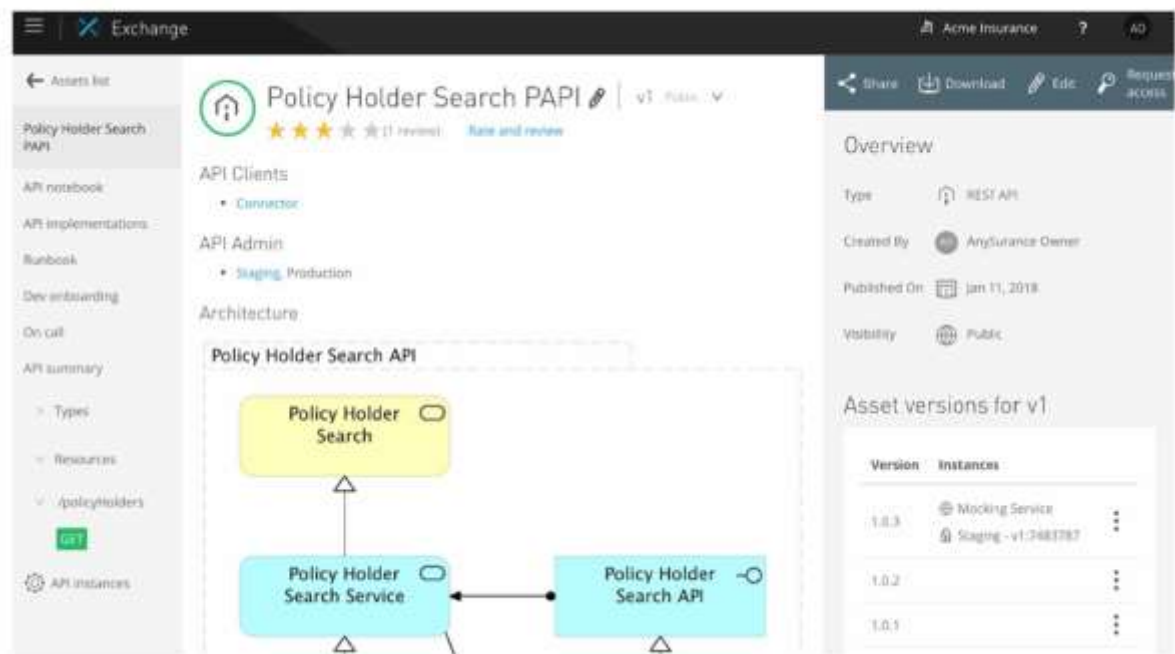


Fig. 3.22 The Anypoint Exchange entry for a particular major version of an API is the portal to this API's documentation and assets.

API implementations typically have described a used component well-defined static described on other APIs and/or backend connect. While similar connect may materialize in Event-Driven approach at runtime, there are not any static dependencies between the appliance components exchanging events. Instead, these application components migration purpose depend upon event the exchanged types, the destinations and therefore the message broker requesting were used hosting those end point. Furthermore, event consumers may change dynamically at any time.

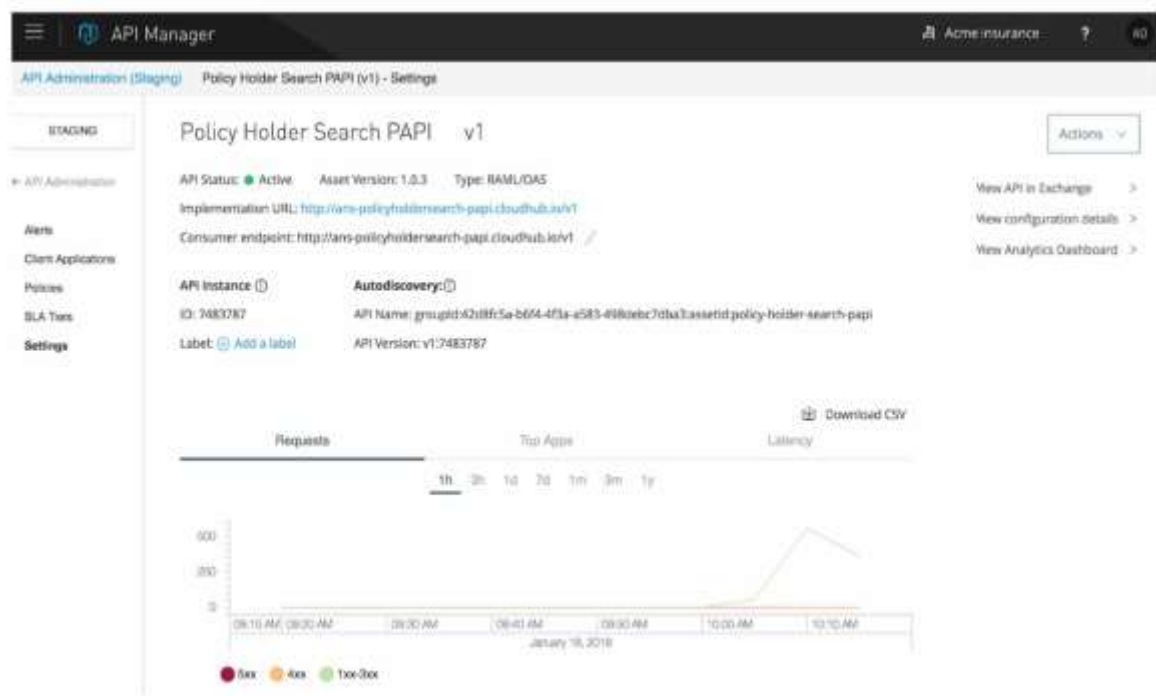


Fig. 3.23 The Anypoint API Manager API administration entry linked-to from its Anypoint Exchange entry, showing summary API analytics.

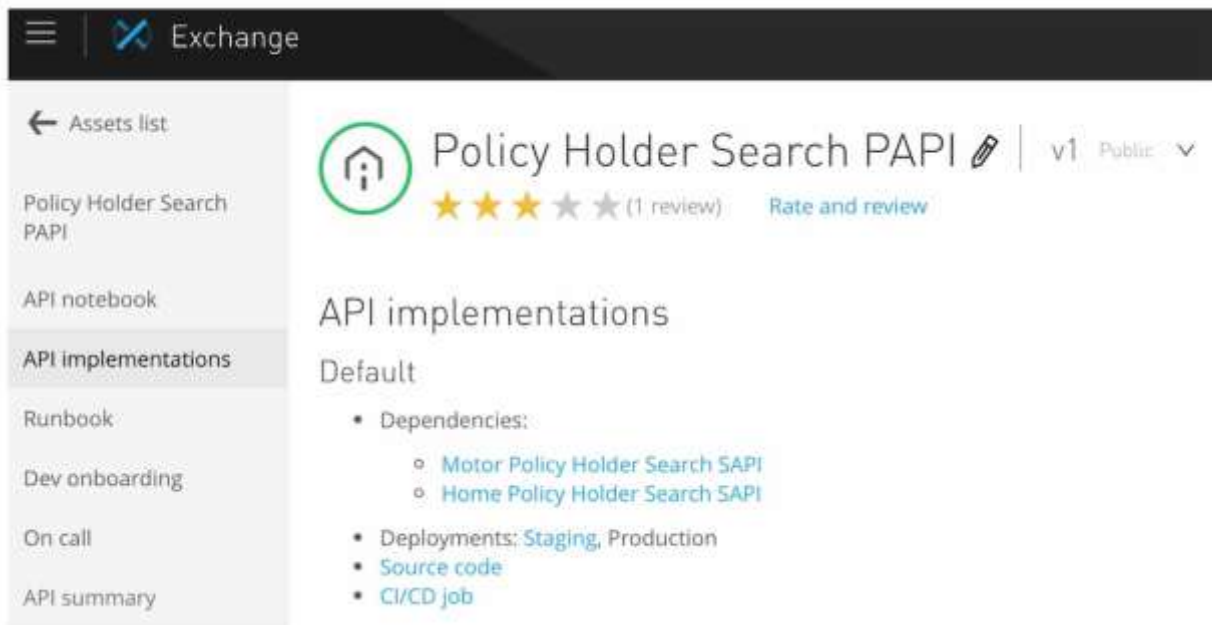


Fig. 3.24 The Anypoint Exchange entry section dedicated to all known API implementations of a particular major version of an API.



Fig. 3.25. The Anypoint Runtime Manager dashboard for the CloudHub deployment of the API implementation linked-to from its Anypoint Exchange entry

Conclusions on the Third Part

This is final implementation part that was described a technical part of application implementation mechanism and it was started with PVN connection setting that is a mandatory part to access data or pushing it to data base. As this part is confidential all nuances were hidden to keep it secure.

Next is general information about setting an Anypoint Studio to development process and include only mandatory parts and can be modified personally by everyone according the preferences.

As the result was described a used components and logic architecture of application. As a deployment part was described a use enterprise approach with deploring it to Anypoint Platform with restricting the data access policy.

CONCLUSIONS

During this work was considered to develop an integration application to synchronize data between two systems. It was decided to use such technology: Mule 4, message queue, data base and private local network. As a result, by requesting from local data base on google calendar side new events is going to be created. This event is possible to update and delete by doing the same request from local data with different parameters. During development was considered a case when creating an any action with events can go wrong so for these purposes were added extra retries approach with notifying Avenga team and user about failed tries.

This application is available connect to any data base by configuring global params in property file that is very flexible approach to set up an application. During the development to avoid outcoming data requesting were used Avenga VPN and also application is validating incoming data for being valid and mail to be corporative Avenga mails.

As the result this is totally free development approach that include Mule 4 community edition that is enough for migration purpose. Mule application is running on MSB that was compered between community edition and enterprise one. Enterprise version is more flexible and easier to deploy and manage from Anypoint Platform that was the main reason Avenga went with this one particular solution, but to make it cheaper deploying to community runtime is also ana option. To go will that solution remote server is required to be configured and have enough memory to start and run an application. One of the popular approaches is using the docker images to deploy it.

To test and basically run an application request from postman is also supported but, in this case, data can possibly be stolen as using the basic http request. To avoid it is recommended to prepare a data base synchronize call to integrate data between systems. To improve this application others data type can be implemented to get outcoming request and

process data asynchronously but it directly will affect on memory required by server to satisfy the operations.

As far as Mule 4 goes with any private modules, it possible to write an your own and implements it directly as proxy or independent part of application exestuation and get benefits from it. Also, as more and more APIs are coming to offer their services it possible to build more powerful data integration service to use more system such as slack to inform a user and team about errors in production.

REFERENCE LIST

1. David Masri. Developing Data Migrations and Integrations with Salesforce: Patterns and Best Practices, 2019 – 112 p.
2. Tom Laszewski. Migrating to the Cloud: Oracle Client/Server Modernization, 2018 – 212 p.
3. Thomas Erl. Cloud Computing: Concepts, Technology & Architecture, 2017 – 42 p.
4. Michael J. Kavis. Architecting the Cloud: Design Decisions for Cloud Computing Service Models (SaaS, PaaS, and IaaS), 2018 – 44 p.
5. Len Bass. Software Architecture in Practice (SEI Series in Software Engineering), 2016 – 112 p.
6. David Dossot. Mule in Action, Second Edition, 2020 – 203 p.
7. Tijs Rademakers. Open-Source ESBs in Action, 2020 – 88 p.
8. Ezra Zygmuntowicz. Deploying Rails Applications: A Step-by-Step Guide, 2018 – 182 p.
9. Alan Beaulieu, Learning SQL: Master SQL Fundamentals, 2017 – 152 p.
10. Положення про дипломні роботи (проекти) випускників Національного авіаційного університету / [уклад. Бойченко С.В., Іванченко О.В.]. – К.: НАУ, 2017. – 63 p.
11. Google main console panel to configure projects here in the below link. [Electronic resource]: URL: <https://console.developers.google.com/apis/dashboard.pdf> (retrieved date: 14.09.2020)
12. Google Calendar API for developing integration with their services in the below link. [Electronic resource]: URL: <https://developers.google.com/calendar.pdf> (retrieved date: 20.09.2020)

13. Testing and studying Google Calendar API in the below link. URL: [Electronic resource]: <https://developers.google.com/calendar/v3/reference.pdf> (retrieved date: 22.09.2020)

14. Mule documentation for http connector and way to use it in the link below. [Electronic resource]: URL: <https://docs.mulesoft.com/http-connector/0.3.9/http-request-connector.pdf> (retrieved date: 24.09.2020)

15. Mule documentation for logger component and way to use it in the link below. URL: [Electronic resource]: <https://docs.mulesoft.com/mule-runtime/4.3/logger-component-reference.pdf> (retrieved date: 14.10.2020)

16. CloudHub Architecture in mule and way to use it in the link below. URL: [Electronic resource]: <https://docs.mulesoft.com/runtime-manager/cloudhub-architecture.pdf> (retrieved date: 01.10.2020)

17. Clustering provides scalability, workload distribution, and added reliability to your applications on CloudHub shown in the link below. [Electronic resource]: URL: <https://docs.mulesoft.com/runtime-manager/cloudhub-fabric.pdf> (retrieved date: 14.10.2020)

18. Mule documentation for connectors and way to use it in the link below. URL: [Electronic resource]: <https://docs.mulesoft.com/connectors/.pdf> (retrieved date: 01.10.2020)

19. MuleSoft Security provides a layered approach to secure your application network in the link below. [Electronic resource]: URL: <https://docs.mulesoft.com/anypoint-security/.pdf> (retrieved date: 07.10.2020)

20. Deploying Mule as a Service to Tomcat in the link below. URL: [Electronic resource]: <https://docs.mulesoft.com/mule-runtime/3.9/deploying-mule-as-a-service-to-tomcat.pdf> (retrieved date: 07.10.2020)