Cooperative Education Report

EPROXC - Effortless Proxy Creation

Mr. Jirayu Promsongwong

School of International & Interdisciplinary Engineering

Major of Computer Innovation Engineering

Faculty of Engineering

King Mongkut's Institute of Technology Ladkrabang

Academic Year 2019

Cooperative Title: EPROXC-Effortless Proxy Creation

Student intern name: Mr. Jirayu Promsongwong

Faculty: Engineering

Department: School of International & Interdisciplinary Engineering

Major: Computer Innovation Engineering

Advisor name: Assistant Prof. Dr.Rathachai Chawuthai

Mentor name: Mr. Maaloot Suprawan

Company: ExxonMobil

Abstract

Application Programing Interface (API) is a software that allows two applications to talk to each other. However, in real-world implementation it is not as simple as its concept. In practice, when APIs are launched, there is usually a huge number of users using APIs. Since there is a high number of users, there is a high probability that the server will have problems. EPROXC or Effortless Proxy Creation is a tool that helps users automatically create an API proxy for the API using MuleSoft products. EPROXC can be used by any user that owns the API and wants to protect it from someone who may wish to attack it or steal data from the API. The creation of an API Proxy using EPROXC can be created without any MuleSoft skills required. EPROXC also provides more security for the API and users can efficiently maintain and monitor their API Proxy.

Acknowledgement

I would like to express my gratitude and appreciation to everyone who has given me the opportunity to be part of the cooperative educational program at ExxonMobil Limited from 15 July 2019 to 10 January 2020. This program has given me countless valuable experiences. This cooperative education report would not be completed if I had not received help and cooperation from the following people:

1. Mr. Vee Tangmanpakdeepong (Manager), Application Integration Manager
2. Ms. Yosita Rungsawat (Supervisor), Staff Development Manager
3. Mr. Maaloot Suprawan (Mentor), Scrum Master Mulesoft API Platform
4. Mr. Nuttachai Chieocharnlikhit, Enterprise intregration Analyst
5. Mr. Tanet Gamchaitavonrattana, Mulesoft Techlead
6. Mr. Natthan Silpanisong, API Platform Analyst
7. Mr. Pranot Mingbunjurdsuk, API Platform Analyst
8. Mr. Kan Kamalanon, API Platform Analyst

I would like to express my gratitude and appreciation to Assistant Prof. Dr.Rathachai Chawuthai, the cooperative education advisor for his brilliant guidance, finding this suitable position, keeping track of my progress and reviewing my report until its completion.

Furthermore, I would like to thank the support staff and other important people, whose names are not included here, who have given me help for both completing this report and adapting to office life.

Jirayu Promsongwong

Table of Contents

List of Figures

Chapter 1

Introduction

This chapter introduces the background, objective, scope, method and expected outcome of the project.

1.1 Background

API or Application Programing Interface has a basic concept: "A software that allows two applications to talk to each other." However, implementing the API in the real-world is not as simple as its concept. When an API is implemented in the real-world, it is not being implementing for only one user because there is huge number of users who will eventually be using the API at any given time, which will cause high traffic. Additionally, in the real-world, not all of the users will be good, because someone may be using it with the intention of attacking the API or stealing data. To solve this problem, we would need something in the middle between the users and the API whose job would be to handle every problem that was aforementioned. Rather than letting users access the API directly, users will access something that will handle everything, this is the concept of the API proxy.

1.2 Objectives

- Offer a tool that reduces time and effort for creating API Proxy
- Offer higher security standard for APIs
- Improve authentication model in organization towards being modern authentication
- Allow developers without MuleSoft experience to have the ability to create API Proxy and use MuleSoft product without prior knowledge
- Centralize all APIs in one place

1.3 Scope

- Create API that allows developers to create API Proxy
- Create a website for ease of API Proxy creation

- Create API Proxy that allows external application to use internal APIs
- Create API Proxy for internal use
- Use modern authentication model to secure APIs

1.4 Method

1.4.1 API (Application Programming Interface)

- Learn about all the methods used in API

1.4.2 Back-end development

- Learn about all the products provided by MuleSoft
- Train and do self-learning about MuleSoft
- Implement API
- Monitor API
- Learn about modern authentication model
- Implement modern authentication for the API using Azure Active Directory
- Store data in the database using Microsoft SQL server

1.4.3 Front-end development

- Learn about React.js framework
- Implement website using React.js framework
- Make a responsive website
- Implement front-end using JavaScript language
- Implement user UI of the application using HTML

1.5 Expected Outcomes

1.5.1 Expected outcomes for ExxonMobil

- Developers can reduce time and effort when creating API Proxy
- Higher security standard for the API through the use of modern authentication concepts

- Centralize all APIs to make it easier for developers to discover and implement the application using APIs
- External application allowing the use of internal APIs
- Ability to manage and track the activity of active and running applications

1.5.2 Expected outcomes for student intern

- Learn how to use MuleSoft products
- Learn CI/CD process
- Improve knowledge on API
- Learn how to use OpenShift and its architecture
- Learn how to implement front-end using React.js framework
- Improve knowledge and develop skills in modern authentication model
- Improve skills on implementing front-end and back-end

Chapter 2

Literature Review

This chapter will discuss the theoretical background, tools that are used in the project and related work, including the old process of creating an API Proxy.

2.1 Theoretical Background

For the implementation of this project, many tools were used. I have divided them into 3 main parts: Front-end, Back-end and Deployment service, and 5 concepts: API (Application Programming Interface), API Proxy, Modern Authentication, OAuth and OAuth On-Behalf-Of flow.

2.1.1 Main methods for front-end

- **React.js** – React.js is a framework that uses JavaScript library to build user interfaces in order to implement a website with the function of linking with the APIs.
- **HTML** – Hypertext Markup Language is a standard markup for webpages that have an element to build blocks of HTML page, whereby all elements are represented by < > tag.
- **CSS** – Cascading Style Sheets is used to describe how HTML elements are to be displayed and to customize the website, which creates ease of use.
- **Unity** – HTML template that is provided by ExxonMobil. Refer to Figure 2.1 in order to see how Unity website looks like.

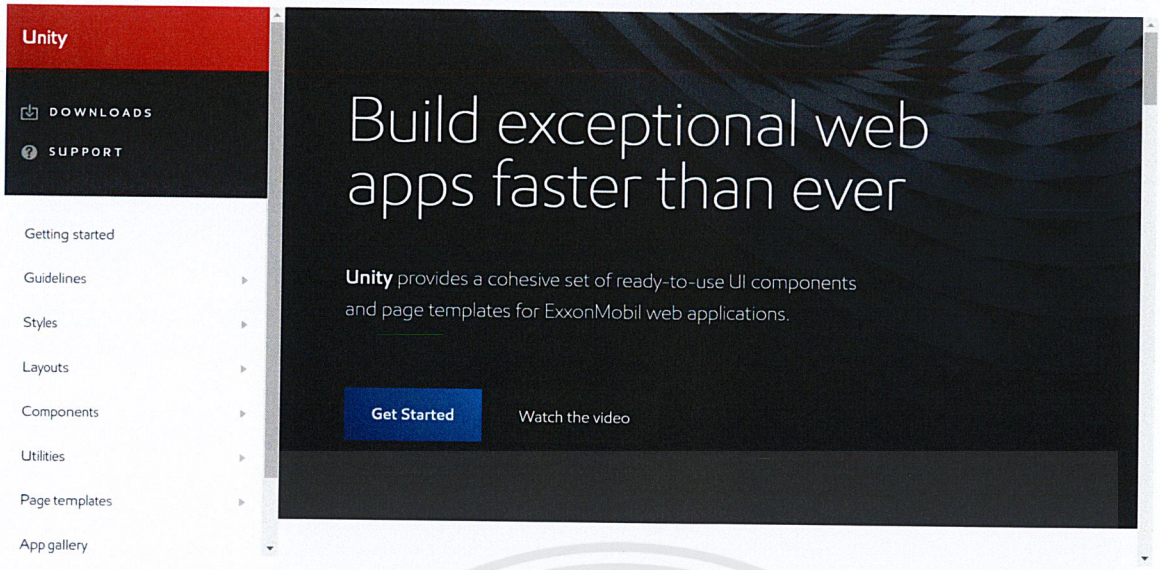

Figure 2.1 Unity website [8]

2.1.2 Main methods for backend

- **MuleSoft –** Product that provides everything the API needs, starting from design to building the API using **"Design Center".** Then the API that was designed or built can be published to **"Anypoint Exchange",** which is the place that developers can search and find MuleSoft APIs in organizations in order to develop their applications. To further develop the API in advance mode, one can use **"Anypoint studio"** and to secure the API one can use **"API manager"** to set some policies to secure the API. Subsequently, to deploy and run the API one can use **"Runtime Manager"**. Finally, to monitor the API one would use **"Anypoint Monitoring"**. MuleSoft is the main product that was used to implement this project.
- **Microsoft SQL Server Management Studio –** To store API and user data in the database to use in front-end and to implement API.
- **Postman –** A tool for testing implemented API.
- **Azure Active Directory –** ExxonMobil is concerned with security, therefore, Azure Active Directory was used to secure API by generating an access token and also validating access tokens to identify users and authorize or check user permission before accessing the API using modern auth model, which are Oauth and On-Behalf-Of flow or OBO flow.

2.1.3 Main methods for deployment service

- **OpenShift –** An open-source container application platform based on the Kubernetes container orchestrator for enterprise application development and deployment. For this project, OpenShift is used to deploy proxy that only allows internal applications to use internal API.
- **Azure DevOps –** A Microsoft product that provides version control, reporting, requirements management, project management, automated builds, lab management, testing and release management capabilities. It covers the entire application lifecycle and enables DevOps capabilities, which are used in this project to control the repositories and also control pipelines for building and releasing the project.
- **Cloudhub -** An integration Platform as a Service (iPaaS) where you can deploy sophisticated cross-cloud integration applications in the cloud, create new APIs on top of existing data sources, integrate on-premise applications with cloud services, and much. This is used to deploy proxy that allows external applications to use internal API.

2.1.4 API (Application Programming Interface)

API is the acronym for Application Programming Interface, which comes with a very basic concept: a software intermediary that allows two applications to talk to each other. For example, when a person uses an application that connects to the Internet and sends data to a server the server will then retrieve that data and perform the necessary actions and send it back to your phone. Following this, the application will interpret the data and present it with the information requested in a readable way. This is an example of an API and everything that happens while using it.

2.1.5 API Proxy

The idea of API proxy was derived from the problems encountered when API is implemented into the real-world, such as, huge amount of users causing high traffic

and problems from users who want to attack or steal data from the API. For this reason, the API Proxy acts on behalf of the front-end of the API and the back-end services to filter all incoming and outgoing traffic, as shown in Figure 2.1. The decoupling of front-end and back-end services allows for changes to be made to back-end services without disrupting the production of API and the filtering of incoming and outgoing traffic allows for monitoring, basic forms of security, requesting routing and protocol translation, which can be set up in the policies of the API Proxy.

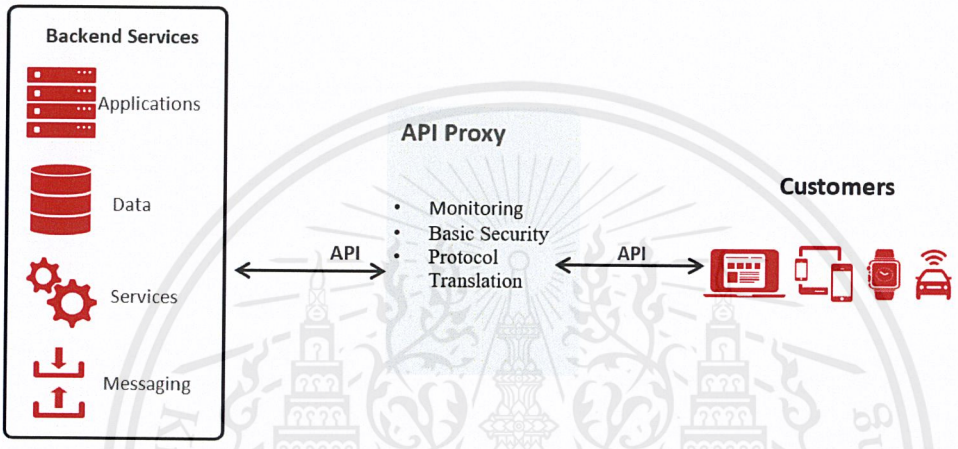

Figure 2.2 Concept of API proxy [3]

2.1.6 Modern Authentication

Method of identity management that offers more secure user **authentication** and **authorization**.

- Authentication

Validating user credentials like Username and Password to verify user identity. Authentication is usually done by a username and password or sometimes with factors of authentication, which refers to the various ways to be authenticated. Authentication is used to verify one's identity for accessing the API.

- Authorization

Authorization determines how much access or permission a user has for accessing the system. Authorization usually comes after authentication, which confirms the user's privileges to perform something in the system.

2.1.7 OAuth

An open-standard authorization protocol or framework that describes how unrelated servers and services can safely allow authenticated access to their assets without actually sharing the initial, related, single login credential. OAuth has the following characteristics: secure, third-party, user-agent, and delegated authorization. For example, when logging onto a website, it can offer more than one way of signing in by using another website or service, such as, Facebook or Google. The user clicks on the button linked to the other website, which provides authentication, and then the original website, that the user is attempting to log-in to, will grant permission based on the authentication gained from the secondary website. The process of OAuth is outlined in the diagram in Figure 2.3.

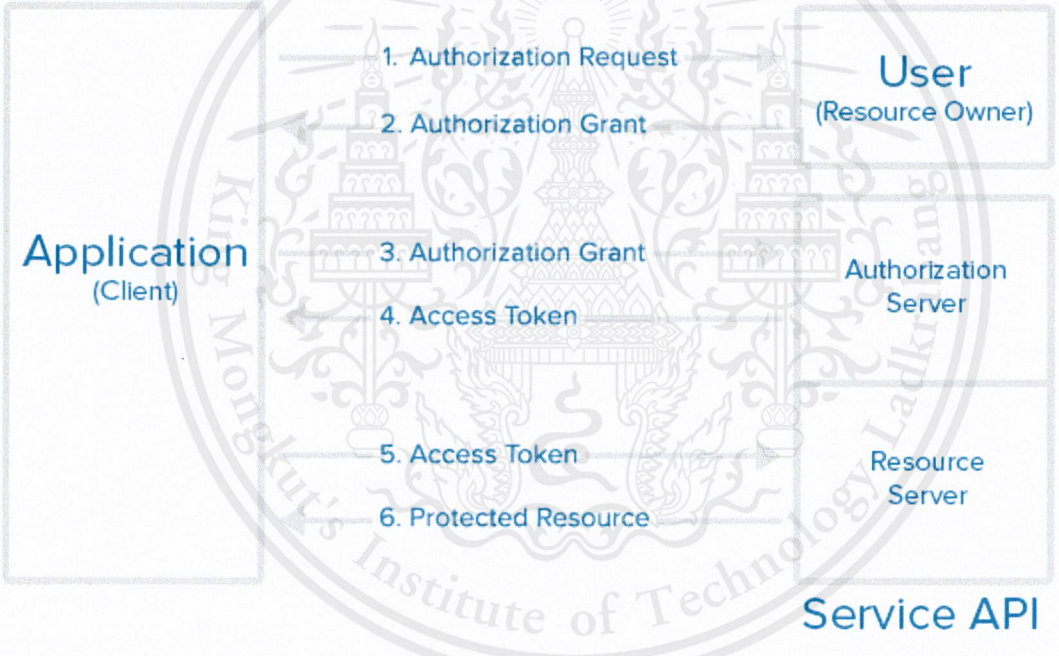

Figure 2.3 OAuth abstract diagram flow [6]

2.1.8 OAuth 2.0 On-Behalf-Of flow

On-Behalf-Of flow or OBO is for cases where an application wants to use another API, which requires user identity not application identity. Thus, in order to transfer user identity from one API to another API, the idea is to propagate the delegated user identity and grant permission through the request chain. In order for the middle-tier service to

request authentication to the downstream service, it needs to secure an access token from the Microsoft identity platform, on behalf of the user, as portrayed in Figure 2.4.

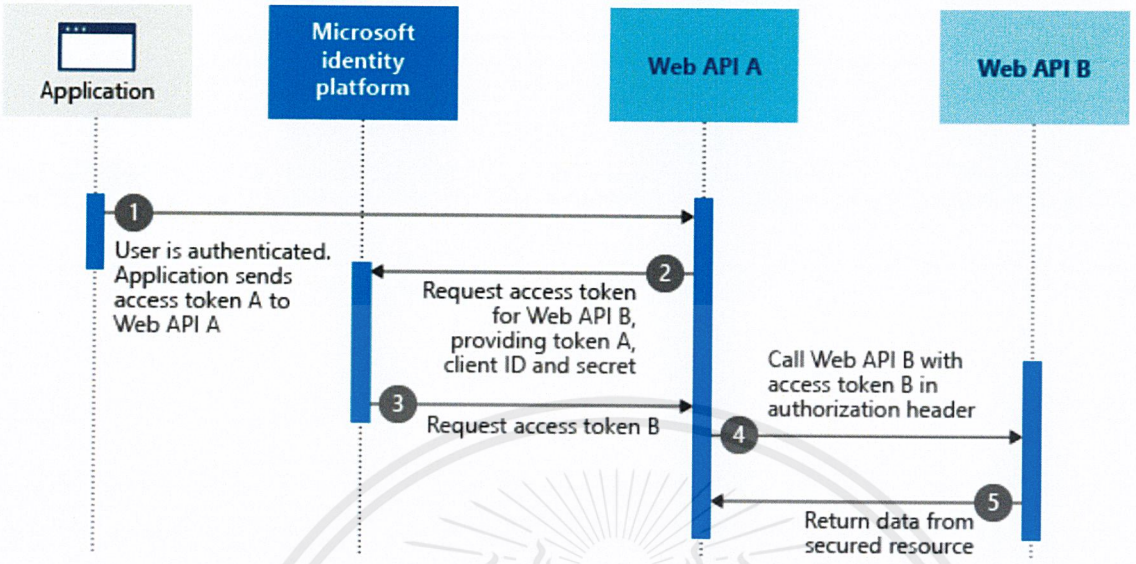

Figure 2.4 On-Behalf-Of flow protocol diagram from Microsoft identity platform [5]

2.2 Related work

The old process of creating API Proxy using Mulesoft is very complicated, takes a lot of time and requires knowledge of Mulesoft, as outlined in Figure 2.5.

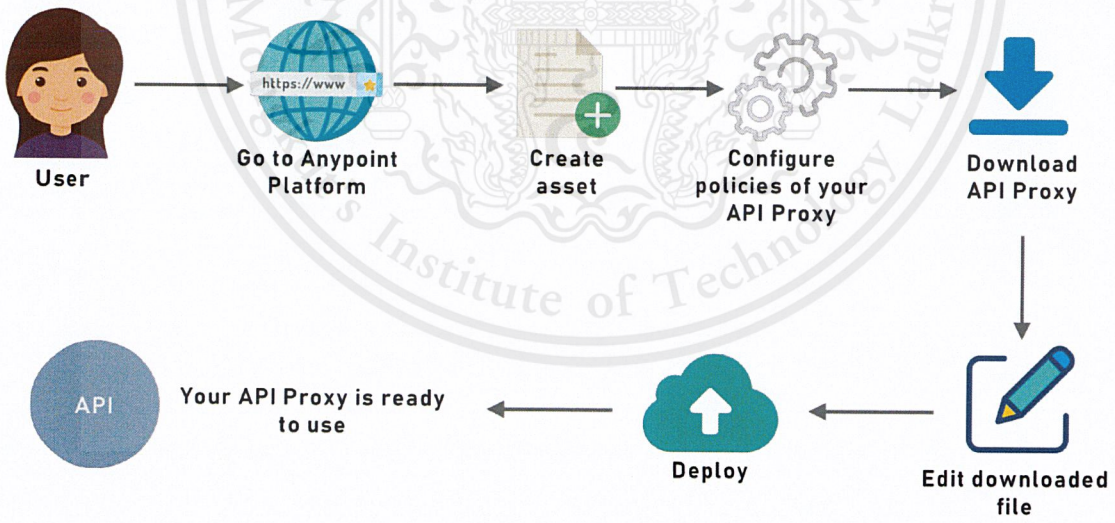

Figure 2.5 Old process of creating API Proxy

2.2.1 Process of creating an API proxy using MuleSoft product

1. Visit Anypoint Platform https://anypoint.mulesoft.com/home/ and the user will see a page like the one displayed in Figure 2.6.

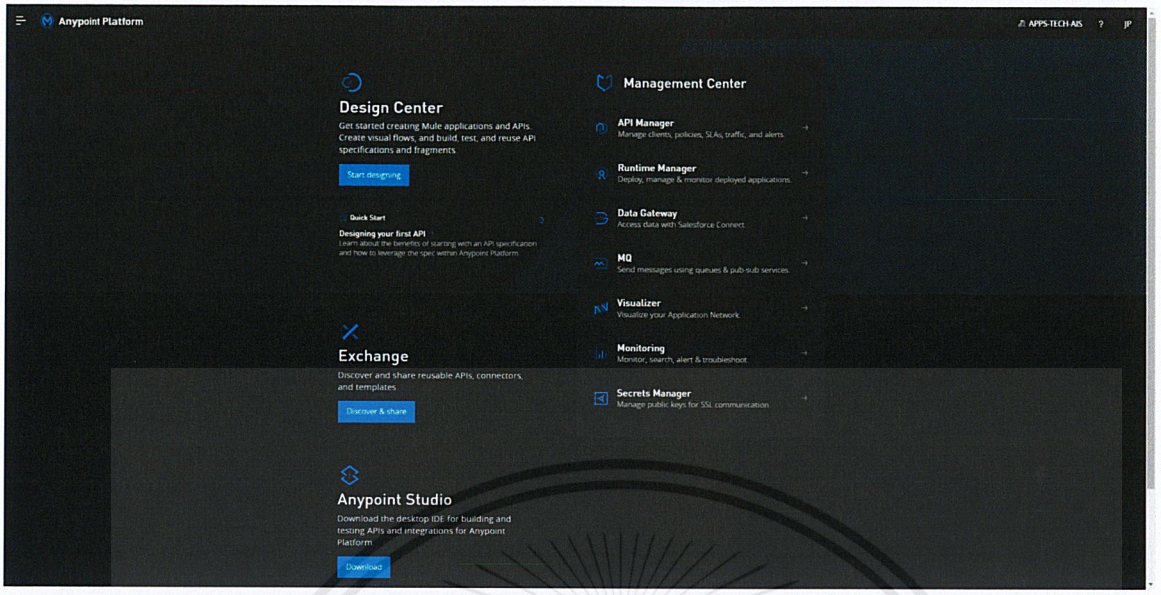

Figure 2.6 Anypoint Platform website [7]

2. When the user clicks on Anypoint Exchange, they will be redirected to a page, identical to the one shown in Figure 2.7. In order to create a new asset for the API proxy, as displayed in Figure 2.8, the user would have to select asset type as being HTTP API.

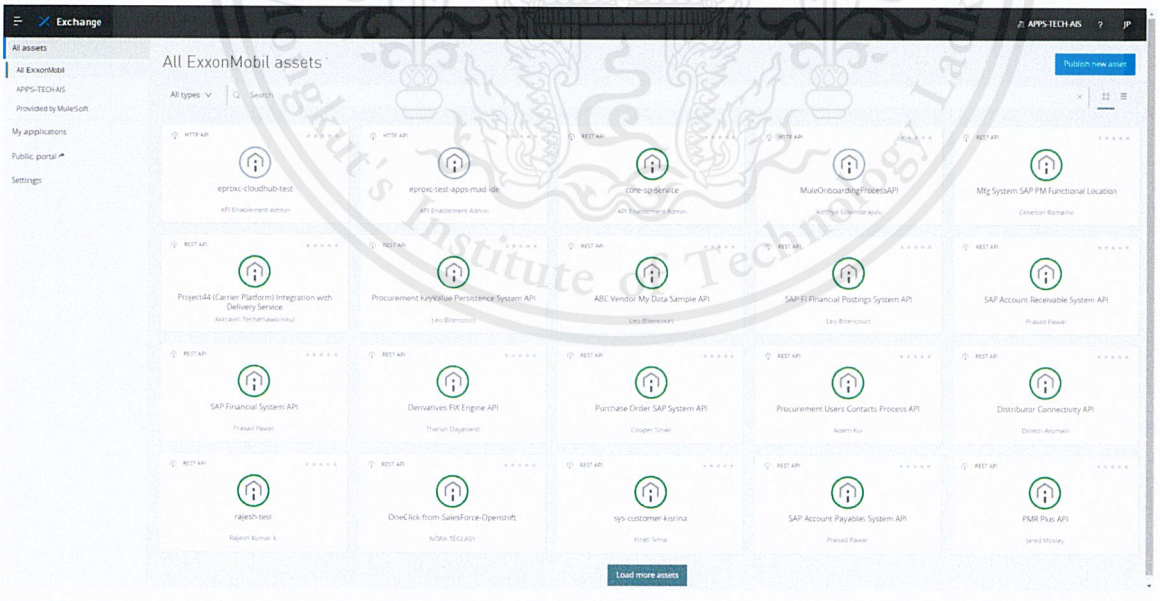

Figure 2.7 Anypoint Exchange page [7]

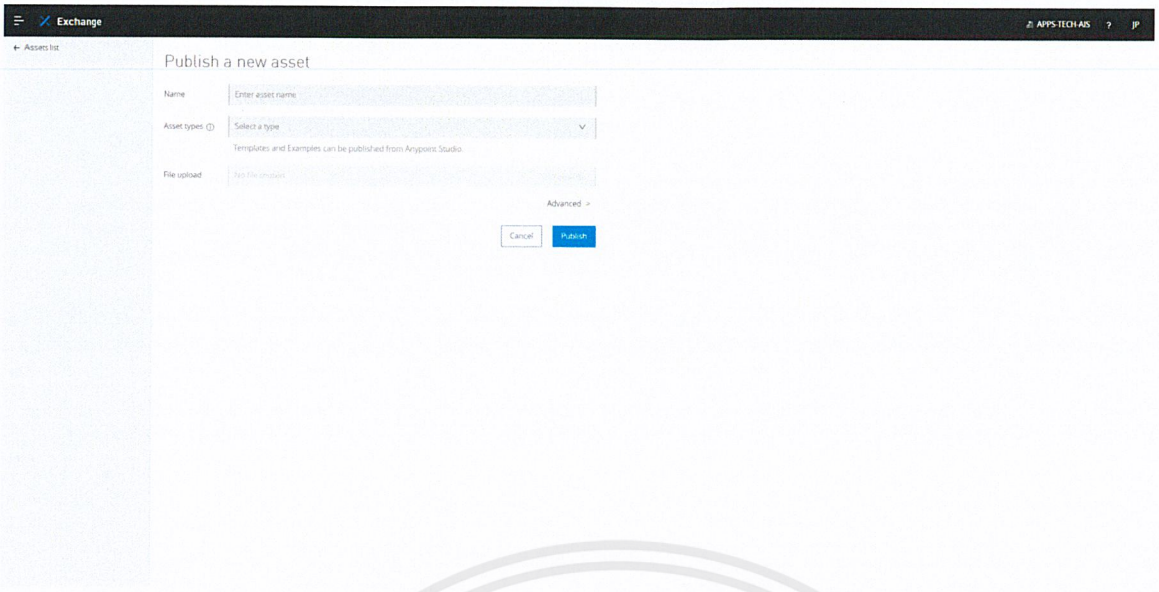

Figure 2.8 Example of creating asset page [7]

3. Once the asset has been created, the user will see an asset page, as demonstrated in Figure 2.9. The user can now write the description, documents or any information of the API proxy. For example, URL, how to use the API proxy? etc.

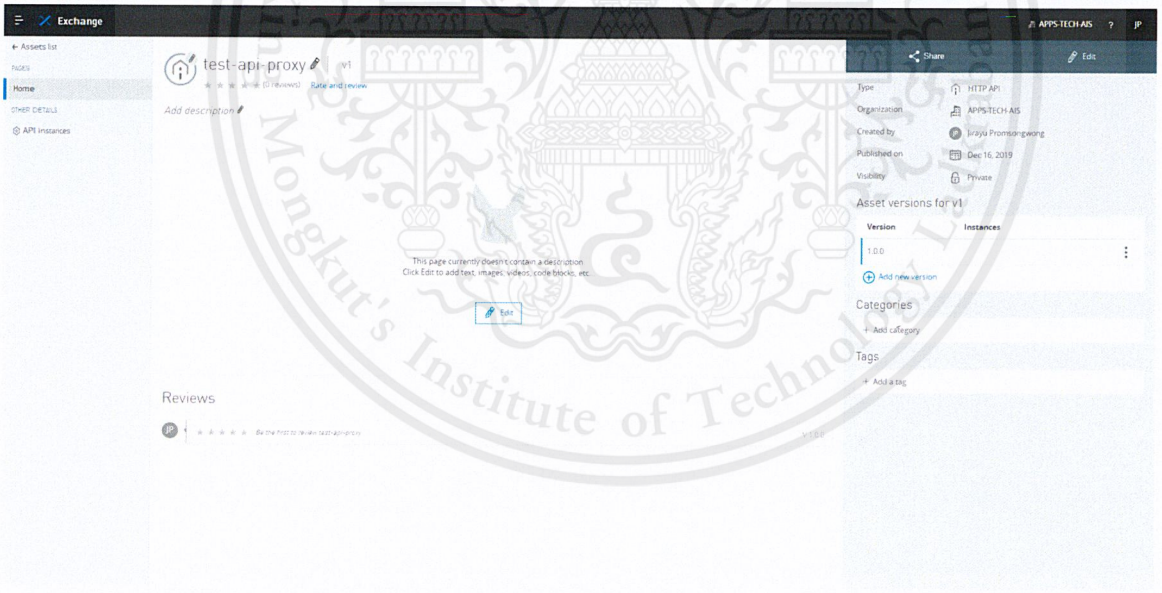

Figure 2.9 Example of asset [7]

4. Once the asset creation process is completed, the user must go to the API manager page, as outlined in Figure 2.10, in order to create an API manager for the API Proxy by clicking on manage API then choosing manage API from Exchange.

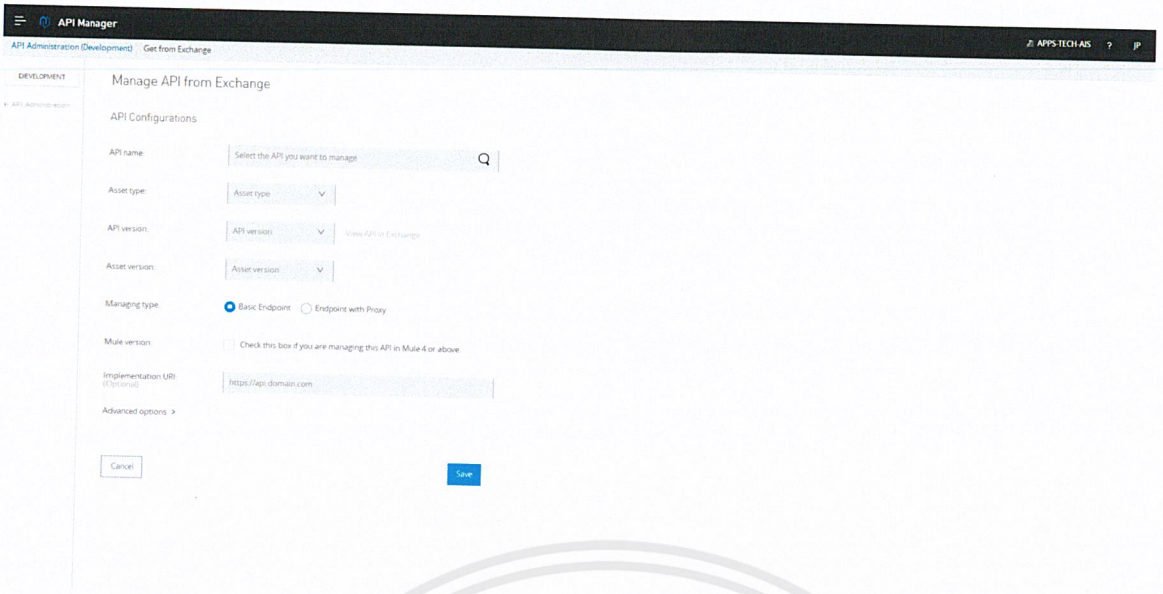

Figure 2.10 Example of configurations page for creating API manager [7]

5. The user must search the asset name, select managing type to Endpoint with Proxy, fill in all required fields and click save.

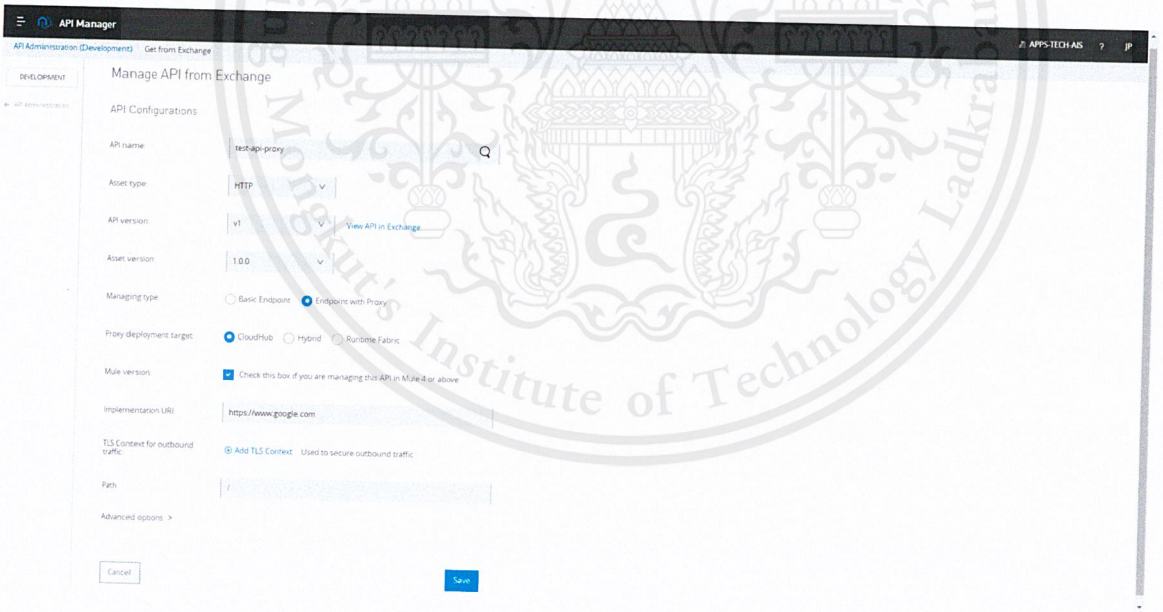

Figure 2.11 Example of configurations page for creating API manager [7]

6. Once created, the user will get an API manager for their API, as shown in Figure 2.12. The user can edit some configurations for the API Proxy or set some policies, for example, Basic Authentication, CORS, Rate limiting, etc. All policies are shown in Figure 2.13.

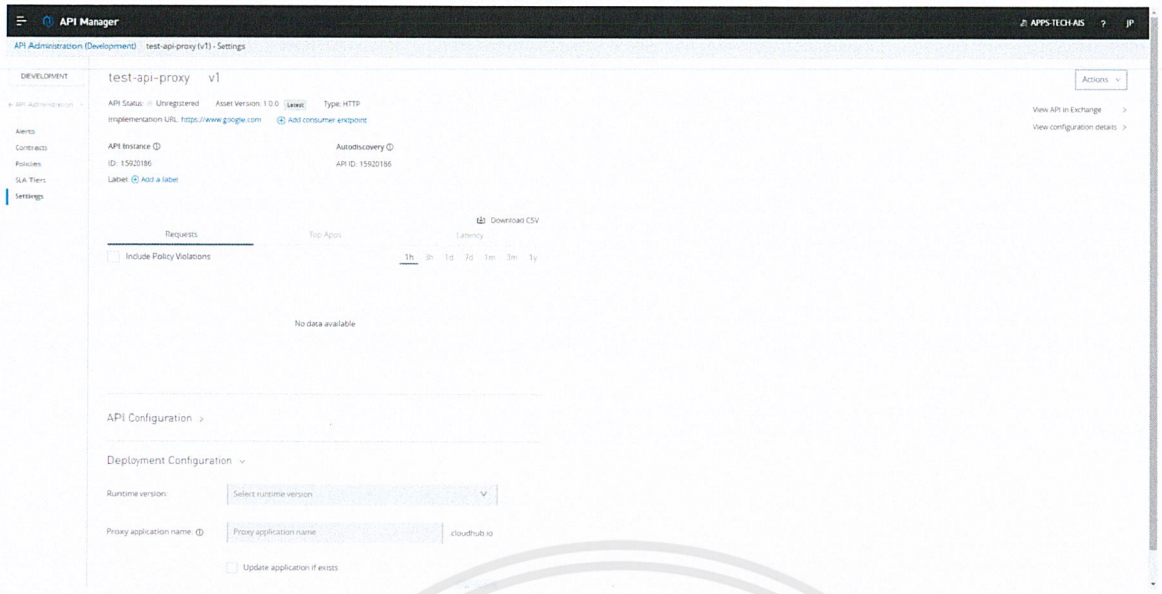

Figure 2.12 Example page of API manager [7]

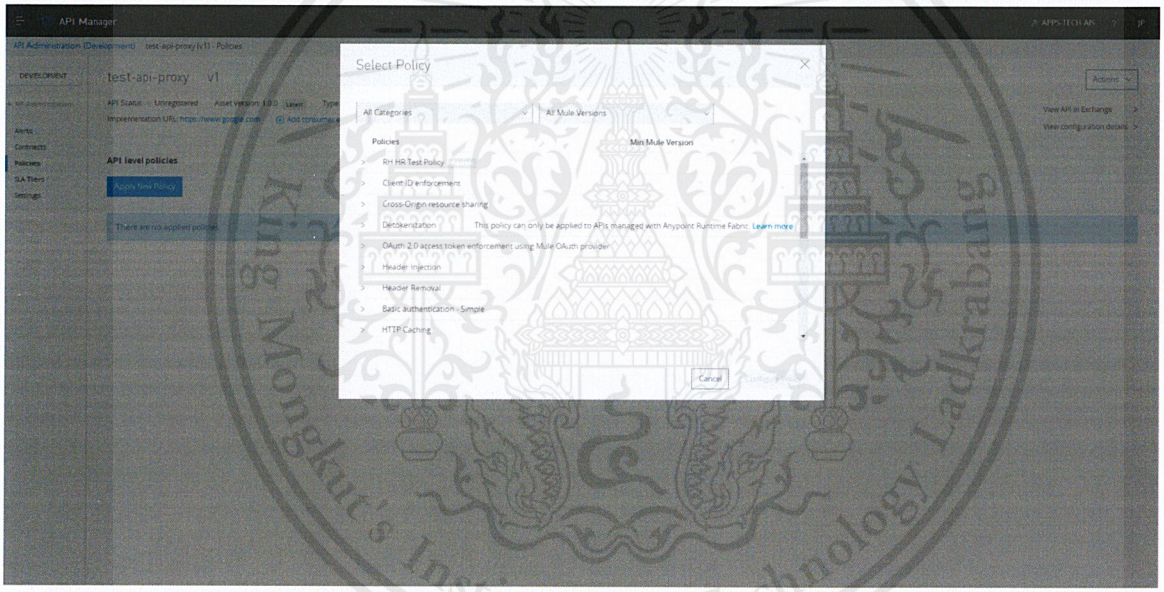

Figure 2.13 Example policies in API manager [7]

7. In order for the API proxy to work with ExxonMobil organization, the user would need to download API Proxy from API manager, edit some configurations using Anypoint Studio and export the file in a Mule deployable archive form, a task which requires knowledge of Mulesoft product.

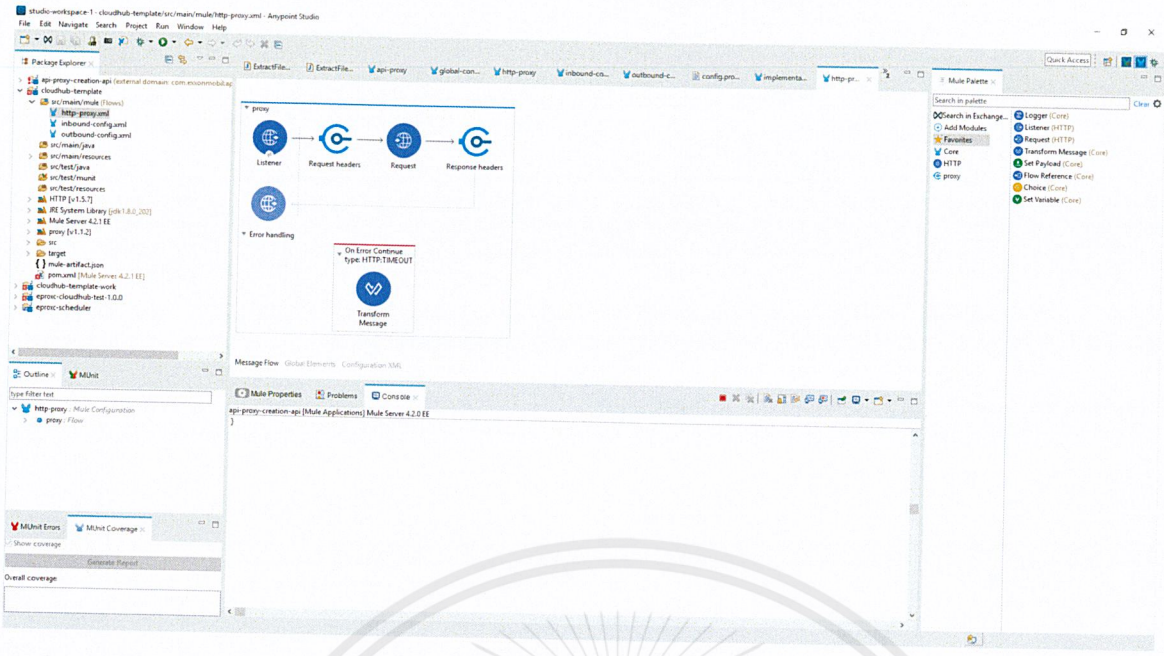

Figure 2.14 Anypoint Studio

8. Once configurations are finalized, the user must go to the runtime manager page, portrayed in Figure 2.15, to deploy the API Proxy.

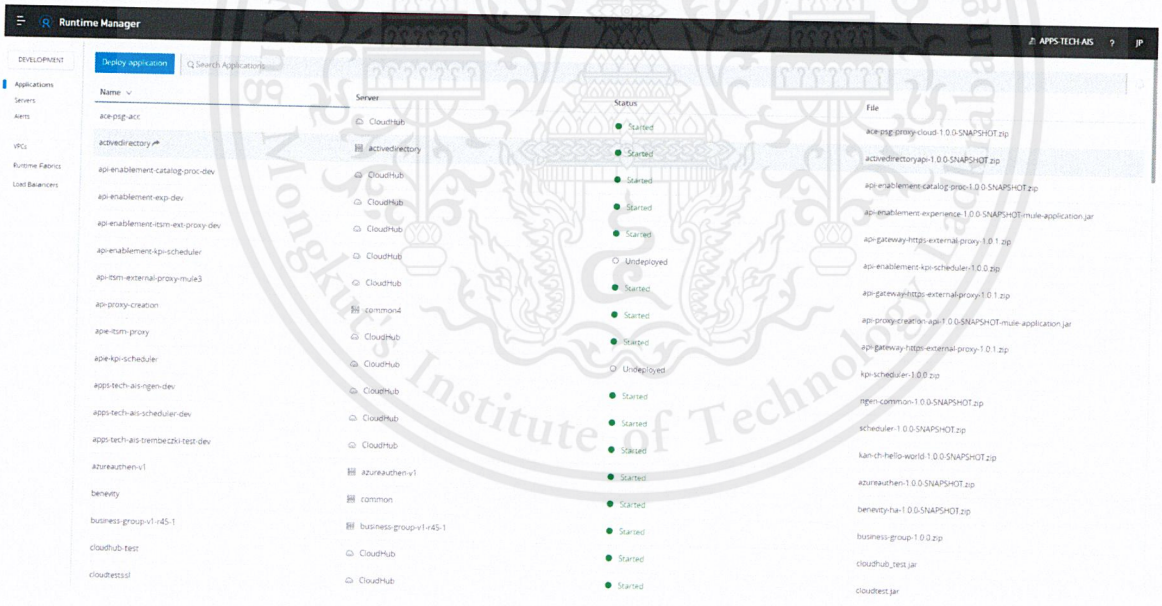

Figure 2.15 Runtime Manager [7]

9. To deploy the API Proxy, the user must select the deployment target or server for the API Proxy to run with and then upload Mule deployable archive file and click to deploy the application.

Figure 2.16 Example page of deploying API Proxy [7]

10. Once deployed, the user will see the runtime manager for the API Proxy, which can be used to monitor traffic of the API Proxy, as outlined in Figure 2.17.

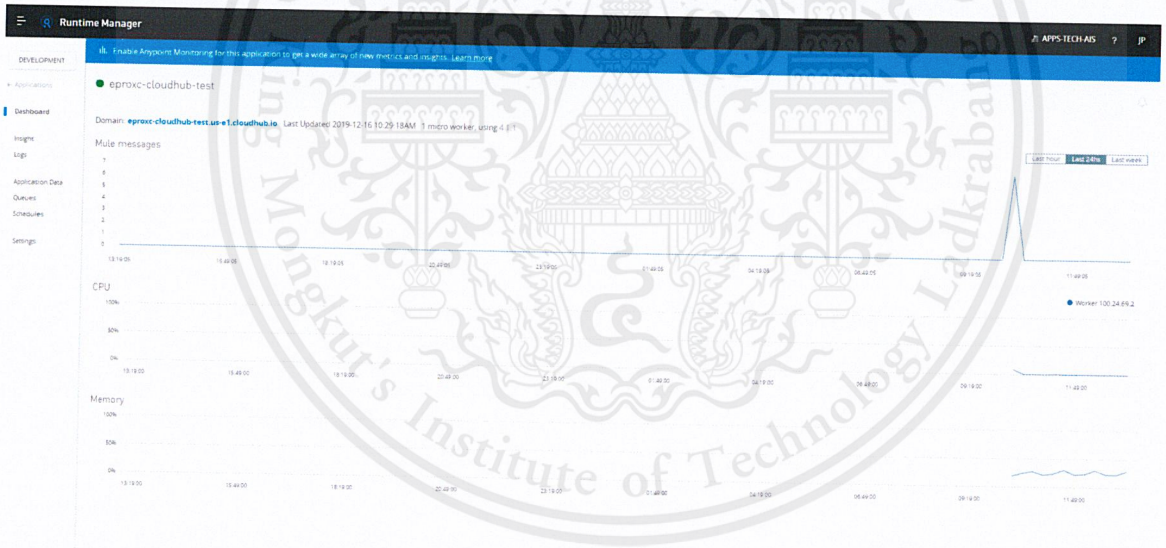

Figure 2.17 Example page of runtime manager [7]

Chapter 3

Methodology

This chapter will provide all the information regarding the project in detail in order to understand the implementation of the project, user stories or user cases, how to deploy the application, security of the application, etc.

The user stories, including each person's requirements, are as follows:

- As an API developer, I want to have a tool that automatically creates an API Proxy for me so that I can reduce time and effort on creating API Proxy

Acceptance criteria:

1. Any tool that can automatically create API Proxy.
2. Reduce time on creating API Proxy.

- As an API developer, I want to track the status of the API Proxy that I created so that I can know its current status.

Acceptance criteria:

1. Status of API Proxy is shown and visible for the user.
2. Status of API Proxy is updated in real-time.

- As an API developer, I want to easily view important information regarding my API Proxy so that when I use the API Proxy I can easily check necessary information.

Acceptance criteria:

1. All necessary information of API Proxy is shown for the user to access and view.
2. All necessary information of API Proxy is easy to access.

- As an API developer, I want to secure my API using API Proxy so that my API will be more secure.

Acceptance criteria:

- User can set policies to secure their API Proxy.

After getting the requirements and reading through all of the user stories, I made a plan to create a tool that would automatically create API Proxy for users by using MuleSoft products but without requiring any MuleSoft knowledge. These are the API for other developers to use in their applications and a website for any user who wants to create an API Proxy for their API and check necessary information of their API Proxy.

Based on the use case diagram in Figure 3.1, EPROXC allows the user to create API Proxy, edit API Proxy, view API Proxy, view API Proxy in their organization, view API Proxy information, delete API Proxy, check API Proxy status, go to Anypoint Exchange and go to API Manager.

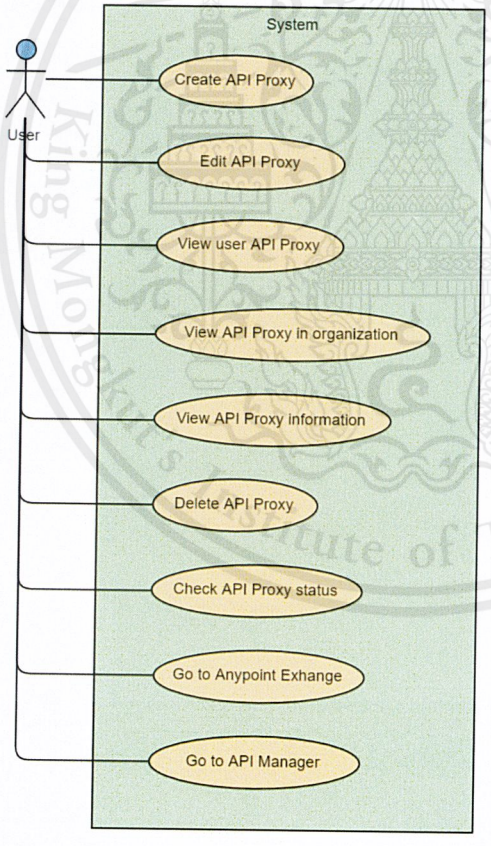

Figure 3.1 Use case diagram of EPROXC

For the back-end part, MuleSoft is used to implement an API that can automatically create an API Proxy.

3.1 EPROXC API

To Implement API, Anypoint Studio is used, which is one of products from MuleSoft that allows the user to implement API in advance. All processes are divided into a flow and to better explain the process of the API, I will explain it flow by flow. However, first one must understand the sequence diagram, as outlined in Figure 3.2 and Figure 3.3, which are the main process of EPROXC API and other cases from Figure 3.1 will be similarly to the main process:

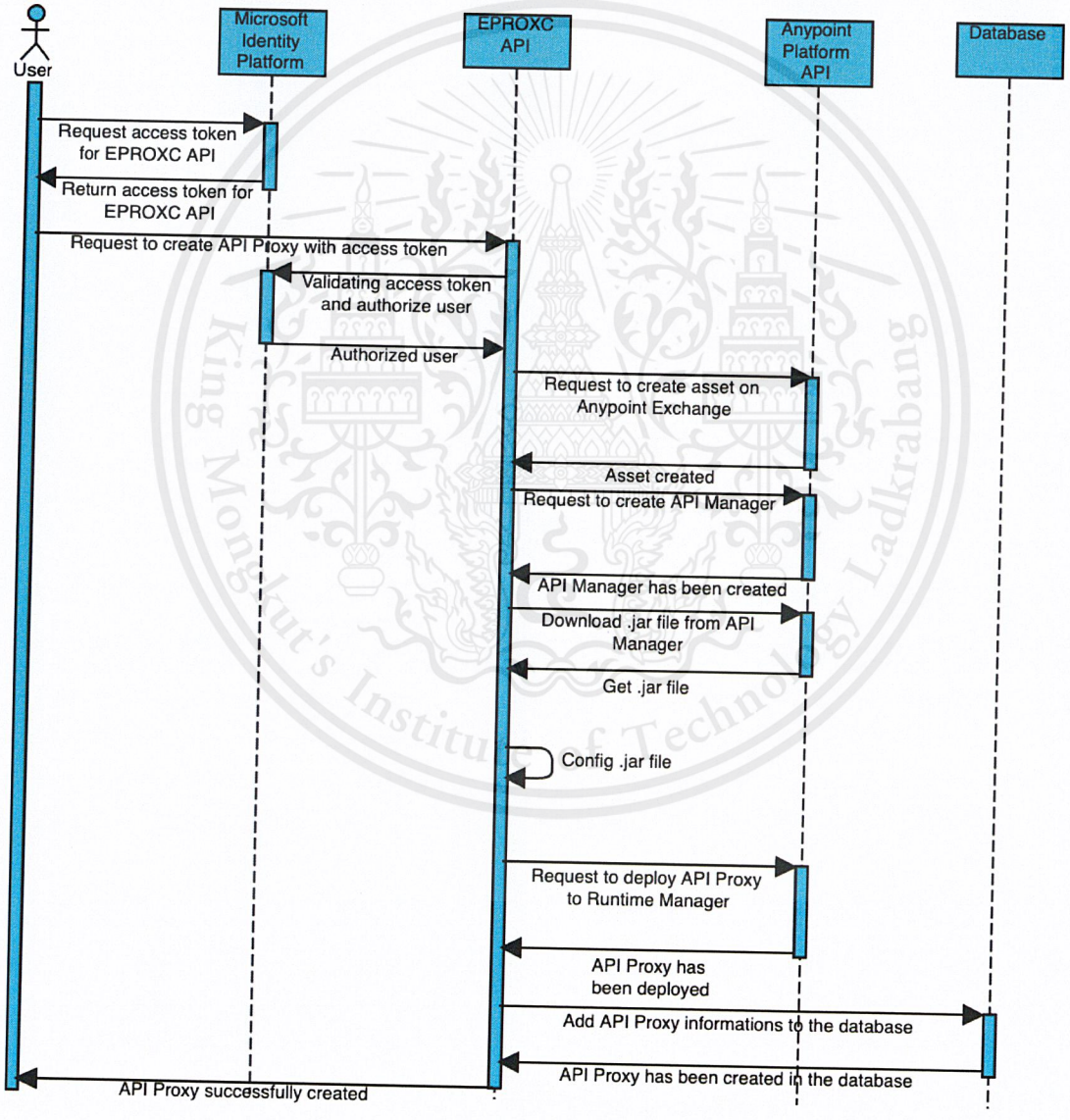

Figure 3.2 Sequence diagram for creating API Proxy of EPROXC API

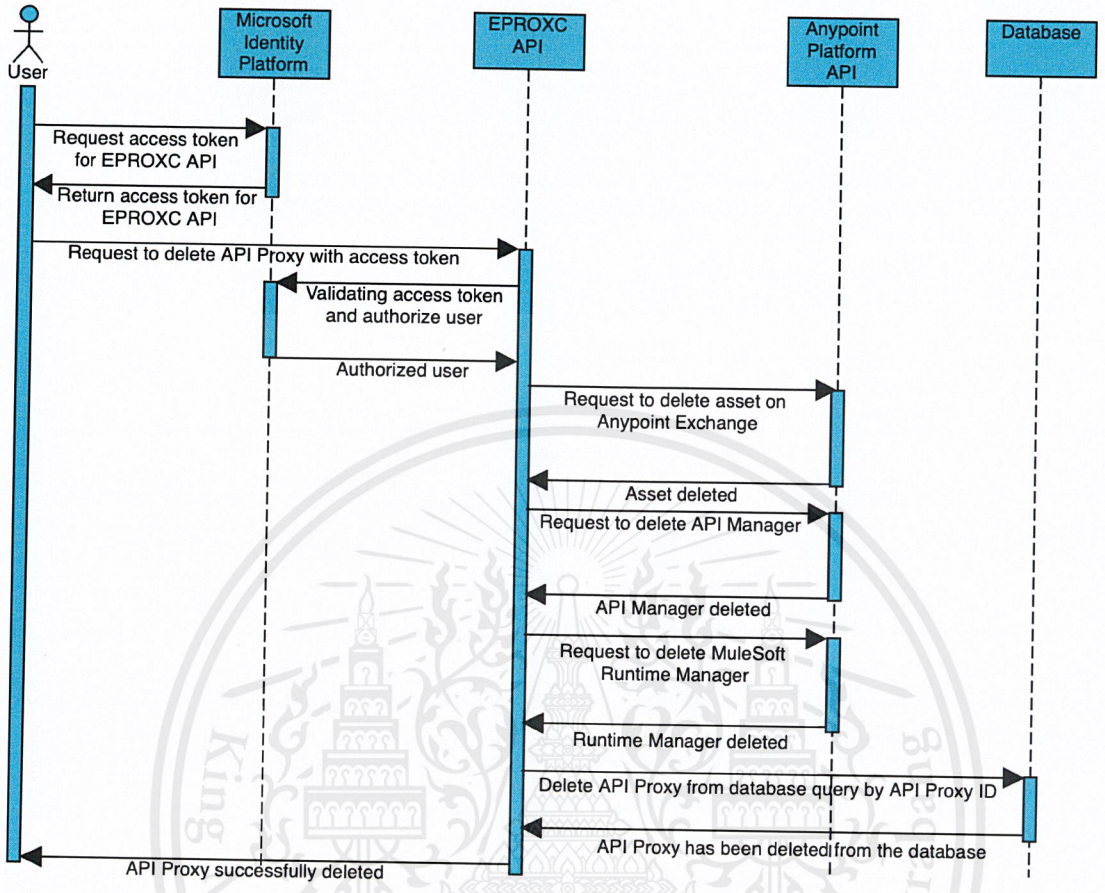

Figure 3.3 Sequence diagram for deleting API Proxy of EPROXC API

We will begin with the endpoint to create an API Proxy

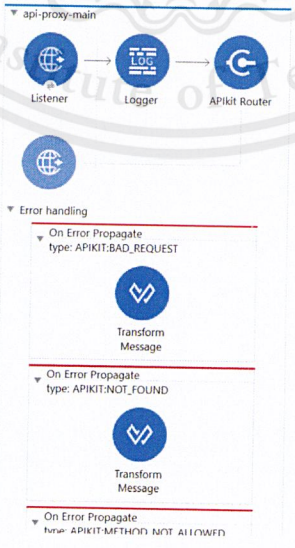

Figure 3.4 api-proxy-main flow

- As shown in Figure 3.4, api-proxy-main flow is a flow that receives all requests from users and leads the requests to the path (user request will follow path name from api-proxy.raml file), which will show all available paths for this API together with the request method, response and required body when users request each endpoint. To find a correct path, APIkit Router is used with api-proxy.raml file. Furthermore, the name of the flow must be in the correct format in order for the APIkit router to find the correct path when requested by the user, otherwise the APIkit Router will not be able to find it.

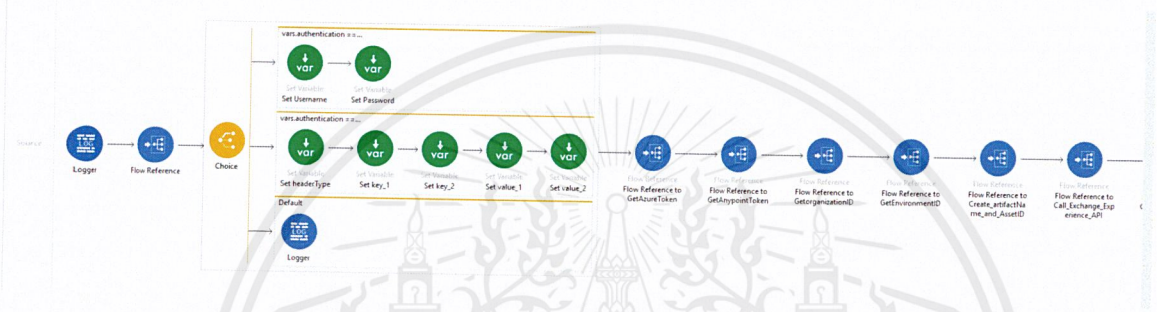

Figure 3.5 Post:\create:application\json:api-proxy-config flow

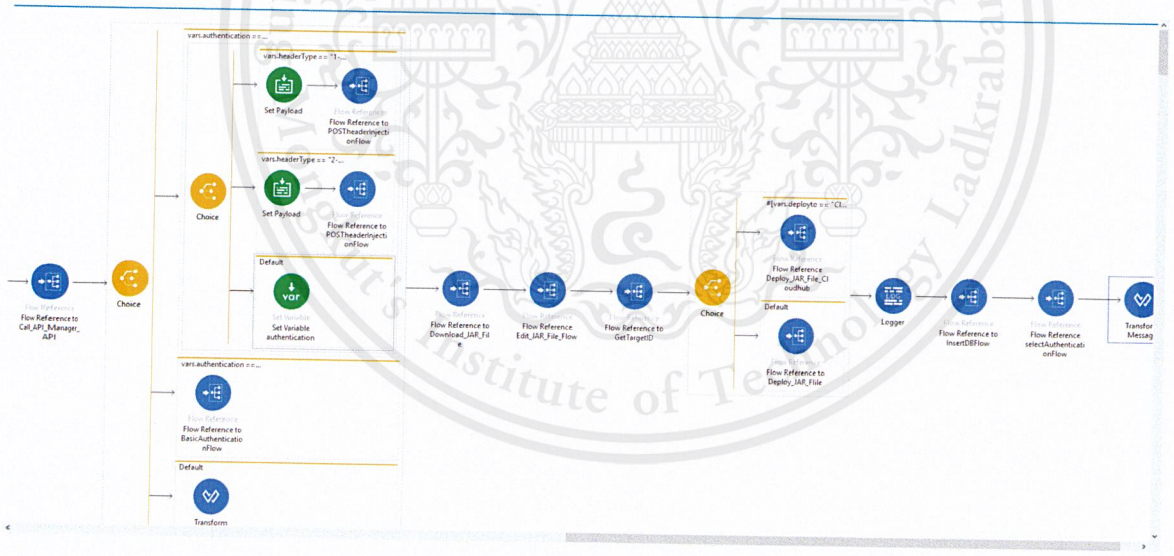

Figure 3.6 Post:\create:application\json:api-proxy-config flow

- As portrayed in Figures 3.5 and 3.6, Post:\create:application\json:api-proxy-config flow is a flow to receive a request using POST method from \create path. The process of this flow is to create API proxy by referencing other flows, basically, this is the main flow of the API.

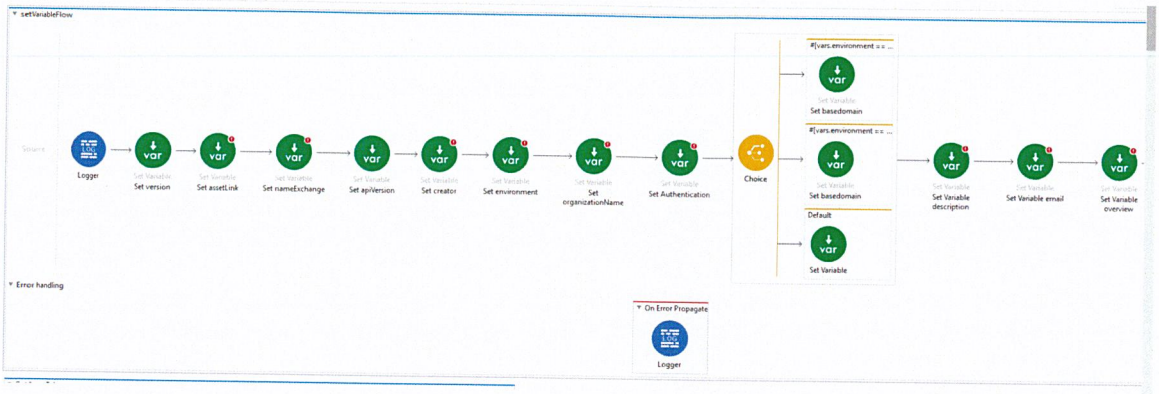

Figure 3.7 setVariableFlow flow

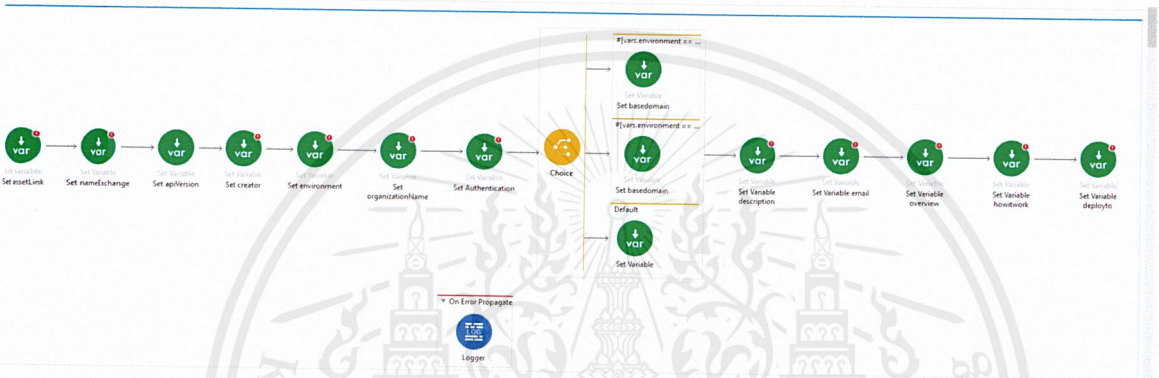

Figure 3.8 setVariableFlow flow

- As displayed in Figures 3.7 and 3.8, setVariableFlow flow is a flow that will collect all JSON content posted from the POST method by a user and set a variable for each component in the JSON body.

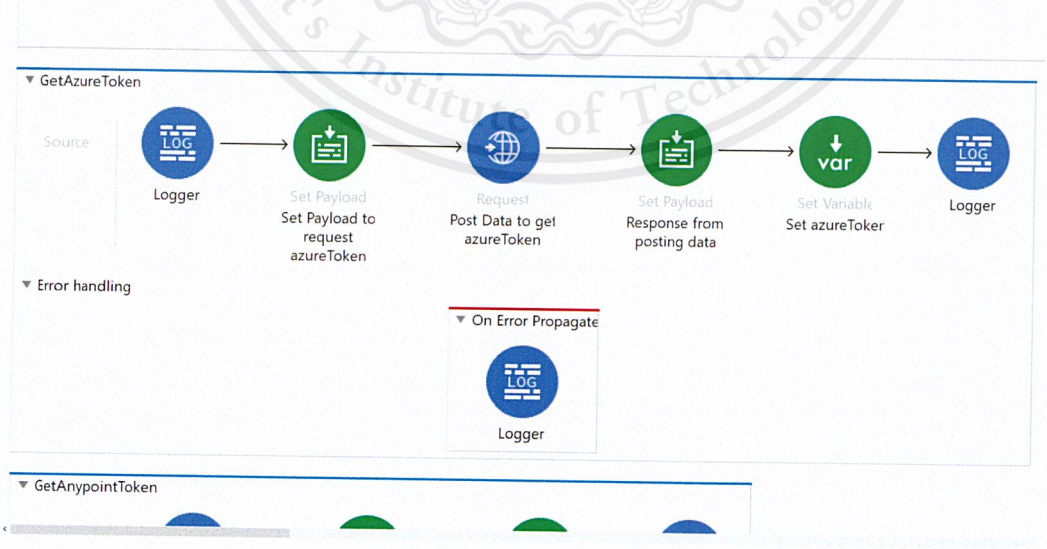

Figure 3.9 GetAzureToken flow

- As outlined in Figure 3.9, GetAzureToken flow uses another API name, "XOM Mule", which is protected by Azure Active Directory using the OAuth model. As shown in Figure 3.9, this flow is to get the access token, which will allow the API to access XOM Mule API to use later. Here, it has been named "Azure token".

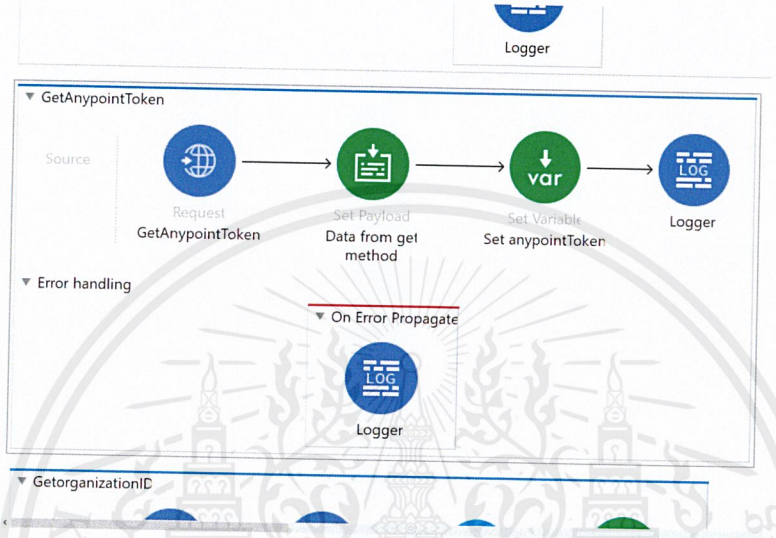

Figure 3.10 GetAnypointToken flow

- Demonstrated in Figure 3.10, GetAnypointToken flow is to create Asset, API manager and Runtime Manager. Anypoint Platform allows API to create those things, but those APIs are protected by using the OAuth model, which means that the user must access the token in order to access the APIs. To get the access token to access Anypoint Platform, XOM Mule API must be used and an access token must be requested for later use. Here, it has been named "Anypoint Token".

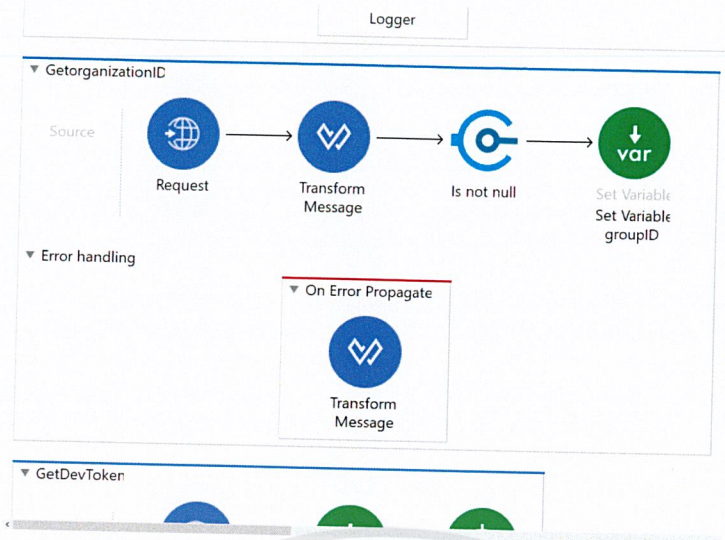

Figure 3.11 GetorganizationID flow

- Figure 3.11 displays the GetorganizationID flow. Several teams at ExxonMobil use Anypoint Platform and each team has an organization name and each organization has their own organization ID. Thus, every API manager and Runtime Manager will sit separately in each organization, however, for the Asset, all of them will sit in one central place on Anypoint Exchange, where any Asset in ExxonMobil can be searched. Furthermore, the user will only be able to edit API manager and Runtime manager in organizations that they have access or permission to edit. To create Asset, API Manager and Runtime Manager, API from Anypoint Platform requires an organization ID. XOM Mule API also provides an endpoint to get organization ID, therefore, this flow is to request XOM Mule API for an organization ID.

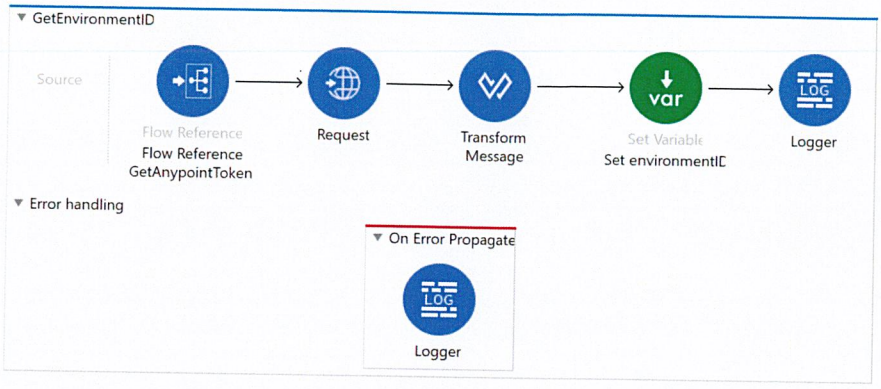

Figure 3.12 GetEnvironmentID flow

- As shown in Figure 3.12, GetEnvironmentID flow is a flow for deploying API manager and Runtime Manager. The user must choose an environment to deploy, from the following:
    - **Development:** Environment for developers to develop API, test or add new features to the API without any effect on API production.
    - **Acceptance:** This environment is nearly in the API production phase but to make sure that the API is ready to deploy to production, the user must first check the Acceptance environment.
    - **Production:** As explained by its name, this environment means that the API has passed all the tests and the API should be stable and ready for production and used by users as a product.

Every environment has its environment ID, which is different in every organization. To identify the environment ID, XOM Mule API provides an endpoint to request the environment ID from any organization in ExxonMobil.

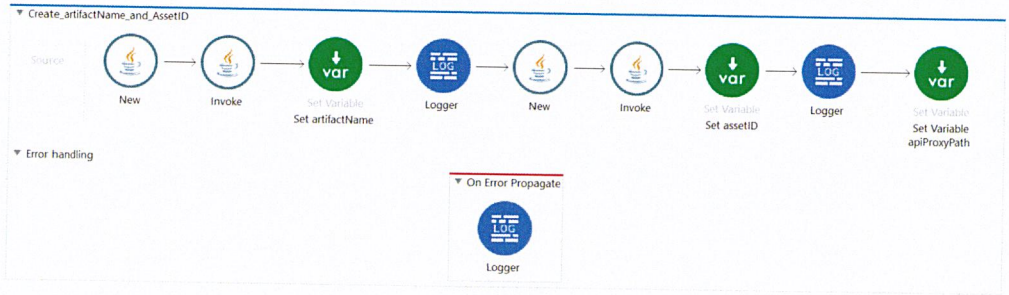

Figure 3.13 Create_artifactName_and_AssetID flow

- As displayed in Figure 3.13, Create_artifactName_and_AssetID flow is to create artifact name and Asset ID from user input by using custom Java functions before creating an Asset on Anypoint Exchange. It also sets URL for API Proxy so that users can use the API Proxy they have created.

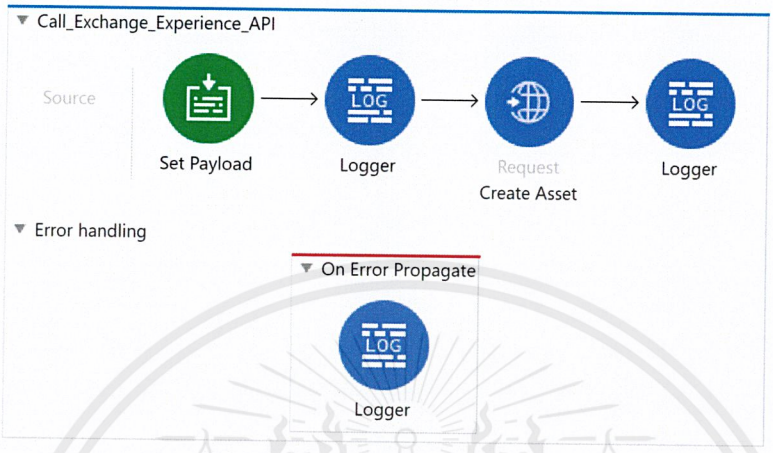

Figure 3.14 Call_Exchange_Experience_API

- As outlined in Figure 3.14, Call_Exchange_Experience_API flow is to create an Asset on Anypoint Exchange. Normally, if the user uses an API that is provided by Anypoint Platform, the Asset will be empty without a template or any information of the API . For this reason, the user must use the API that is created by a developer in ExxonMobil on Anypoint Exchange API named "Catalog API", which is for templating the Asset in order to show the description, details and document of the API.

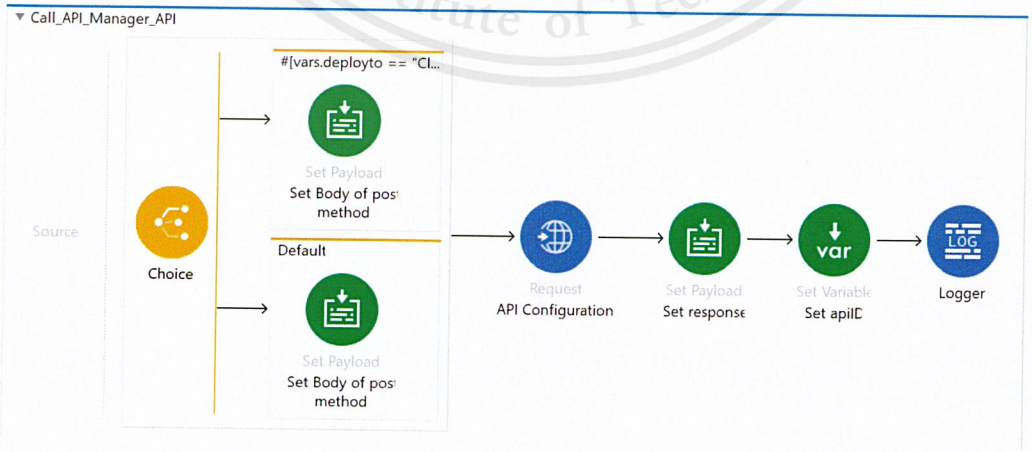

Figure 3.15 Call_API_Manager_API flow

- Figure 3.15 shows the Call_API_Manager_API flow, which is to create an API Manager. Here, an API from Anypoint Platform was used to create an API Manager and once created, the API ID will be available. Subsequently, it can be used to verify API in other flows.

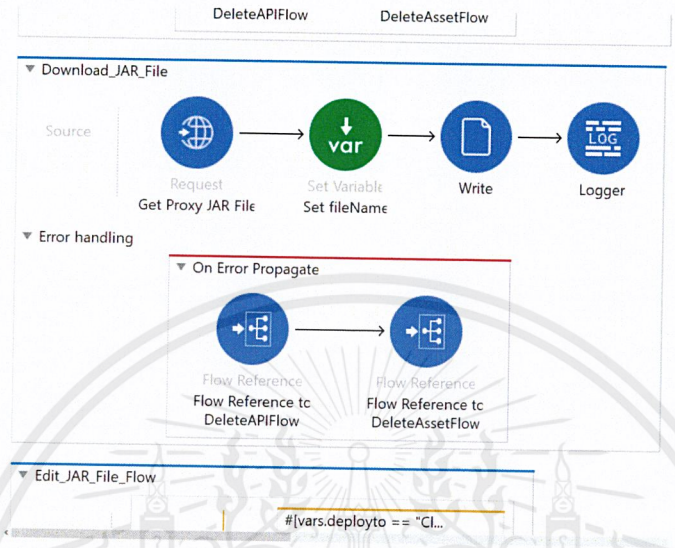

Figure 3.16 Download_JAR_File flow

- As portrayed in Figure 3.16, Download_JAR_File flow is to make the API Proxy work inside the ExxonMobil organization. The file must be edited to pass all security measures in ExxonMobil. However, before the API Proxy file can be edited, it must first be downloaded in the Anypoint Platform. This flow is to call the API and download the API Proxy file.

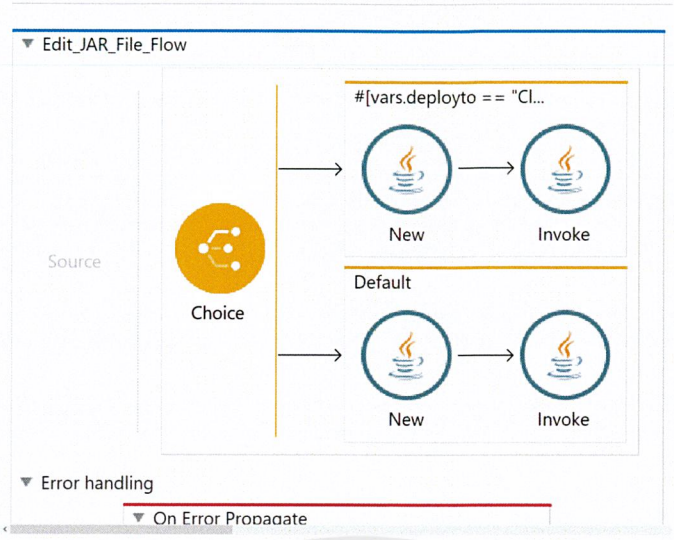

Figure 3.17 Edit_JAR_File_Flow flow

- Figure 3.17 outlines the Edit_JAR_File_Flow flow. Once the API Proxy file has been downloaded, the file can be edited using Java function with the template created in the project folder. Here, the templates have been divided into Openshift and Cloudhub. OpenShift aligns all API Proxy into the same format, which is suitable for internal use, inside ExxonMobil. On the other hand, Cloudhub has a different configuration since Cloudhub allows an external application to use internal API, which means a dummy certificate must be added in order to certify API Proxy.

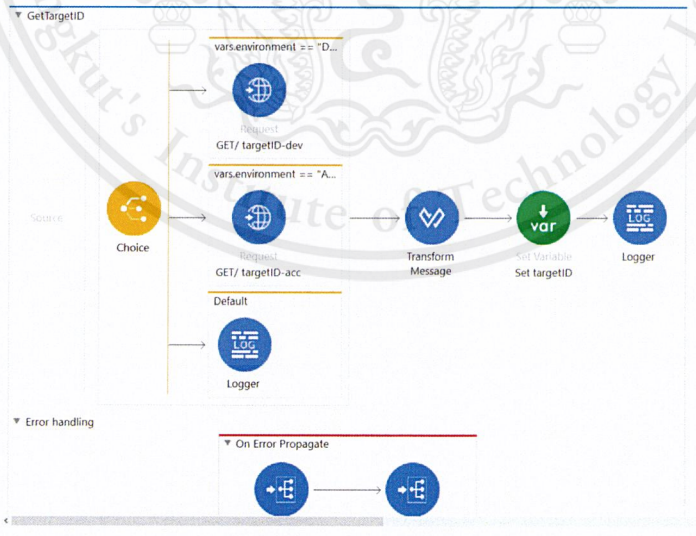

Figure 3.18 GetTargetID flow

- As shown in Figure 3.18, GetTargetID flow is to deploy API Proxy to OpenShift. Here, endpoint from Anypoint Platform API is used, which requires target server ID to

deploy API Proxy to that server, therefore, this flow is for getting target server ID from Anypoint Platform API.

Figure 3.19 Deploy_JAR_FIile flow

- As displayed in Figure 3.19, Deploy_JAR_FIile flow is to deploy the edited API Proxy file to Openshift on Runtime Manager.

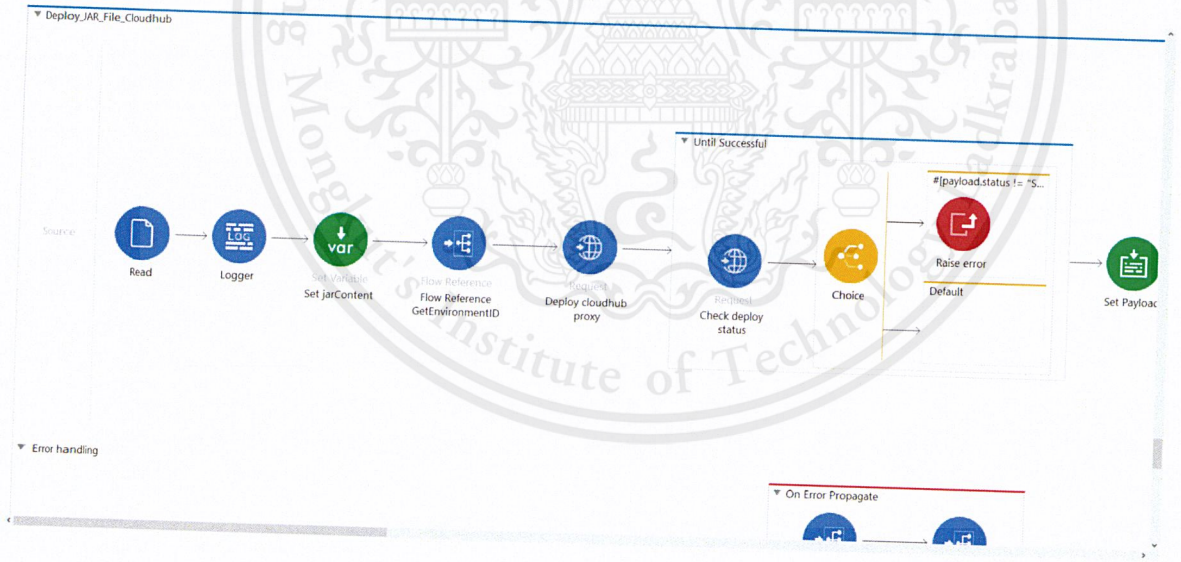

Figure 3.20 Deploy_JAR_File_Cloudhub flow

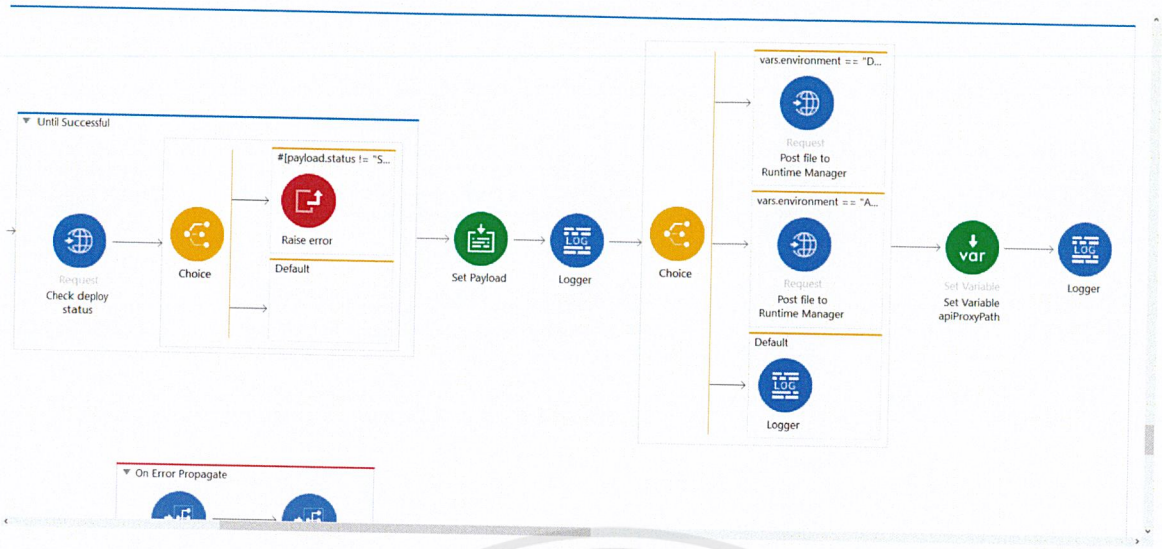

Figure 3.21 Deploy_JAR_File_Cloudhub flow

- As outlined in Figures 3.20 and 3.21, Deploy_JAR_File_Cloudhub flow is to deploy an edited API Proxy file to Cloudhub on Runtime Manager.

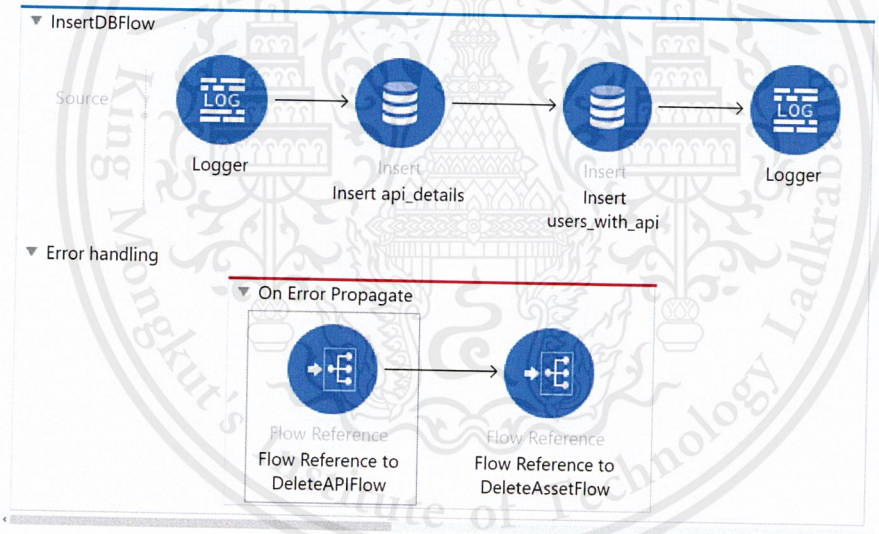

Figure 3.22 InsertDBFlow flow

- As demonstrated in Figure 3.22, InsertDBFlow flow is to store all API Proxies information created using EPROXC in a Database using Microsoft SQL Server. This is to be used in the EPROXC website so users can view their API Proxy.

Security for API Proxy

To secure the API Proxy, EPROXC provides a simple authentication procedure, known as Basic Authentication, which includes The Client ID and Secret. The following flows are used to implement these authentications:

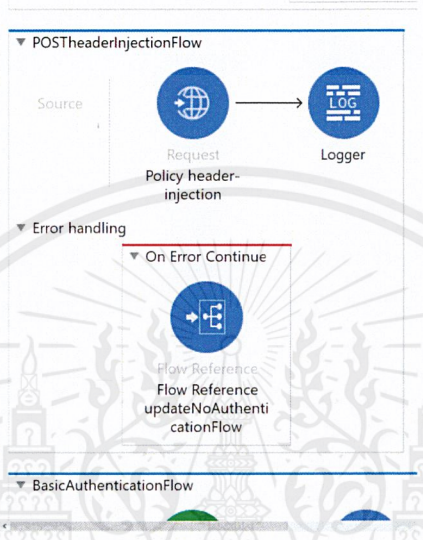

Figure 3.23 POSTheaderInjectionFlow flow

- As shown in Figure 3.23, POSTheaderInjectionFlow flow is to set policy in API manager to inject Client ID and Secret to the header of the request, from the API Proxy and from the API for the authentication process.

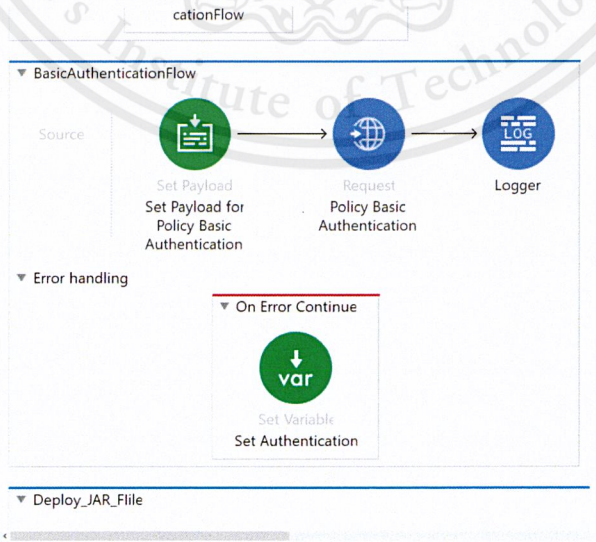

Figure 3.24 BasicAuthenticationFlow flow

- As displayed in Figure 3.24, BasicAuthenticationFlow flow is to set policy in API manager so that the API Proxy will be protected by using Basic Authentication with username and password.

Additional features in EPROXC:

1. Endpoint with GET method to retrieve API information from the database:

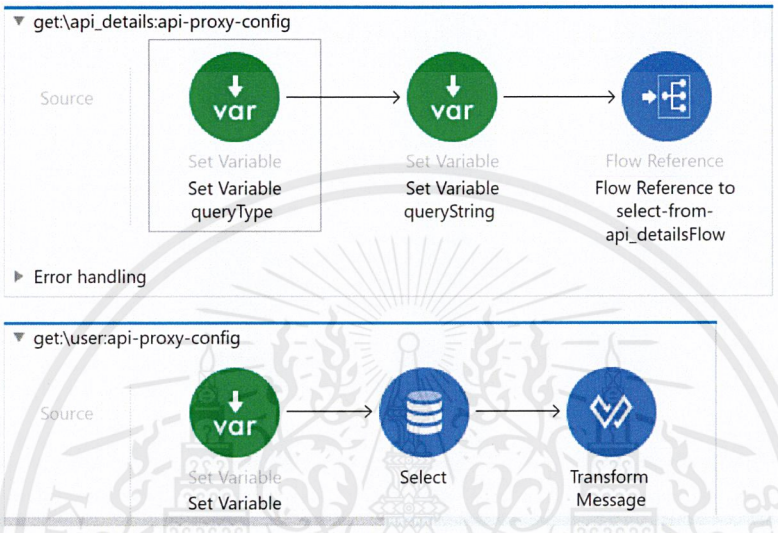

Figure 3.25 get:\api_details:api-proxy-config flow

- As portrayed in Figure 3.25, get:\api_details:api-proxy-config flow is to receive the request from a user to get API Proxy information that has been created or API Proxy in each organization depending on query type and query sting that is requested. Subsequently, the data will be retrieved from the other flow name "**select-from-api_detailsFlow flow**", shown in Figure 3.26. The API Proxy information will then be returned to the user.

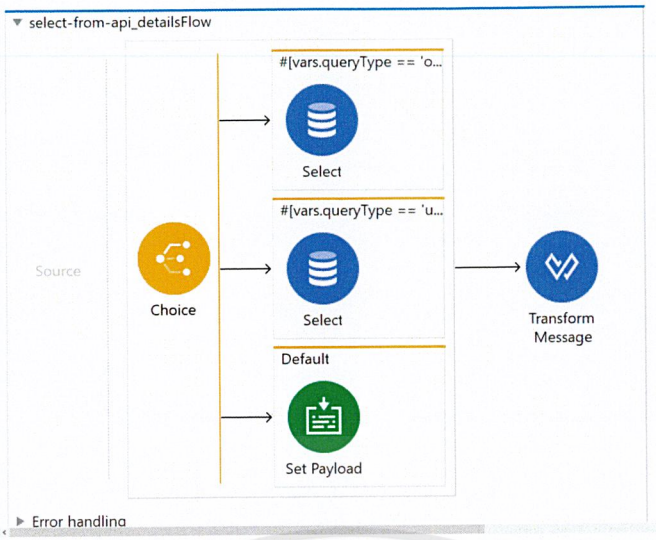

Figure 3.26 select-from-api_detailsFlow flow

- As outlined in Figure 3.26, select-from-api_detailsFlow flow is to query data from the database based on query type and query string of the request.

2. Endpoint with PATCH method to patch the API Proxy. EPROXC allows a user to patch or update their Client ID and Secret, change URL of the API or add Client ID and Secret for API Proxy that does not have any authentication.

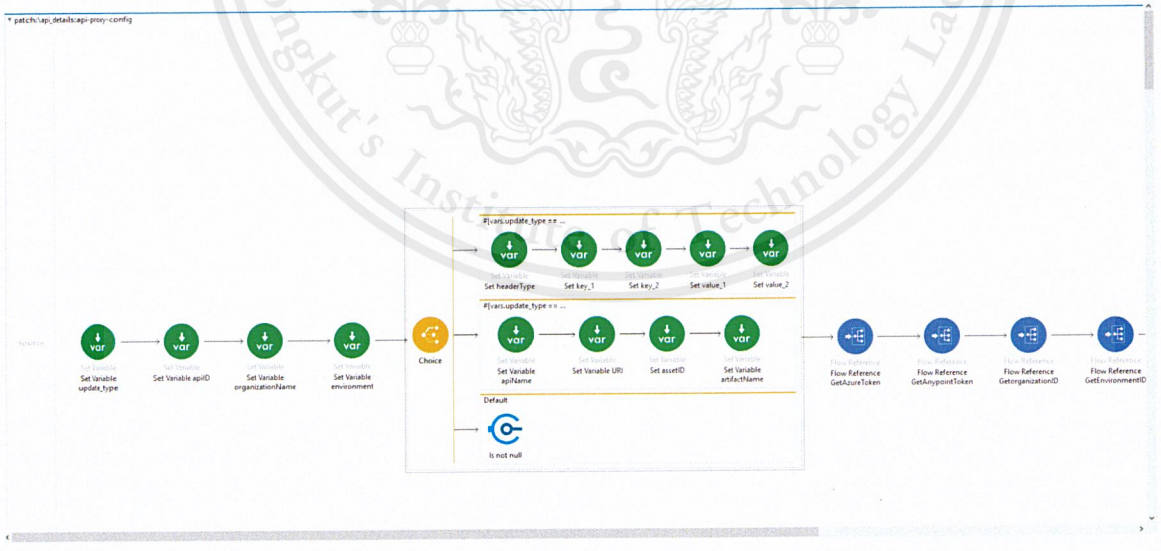

Figure 3.27 patch:\api_details:api-proxy-config flow

Figure 3.28 patch:\api_details:api-proxy-config flow

- As shown in Figures 3.27 and 3.28, patch:\api_details:api-proxy-config flow is to receive the patch request process depending on the update type from the request body.

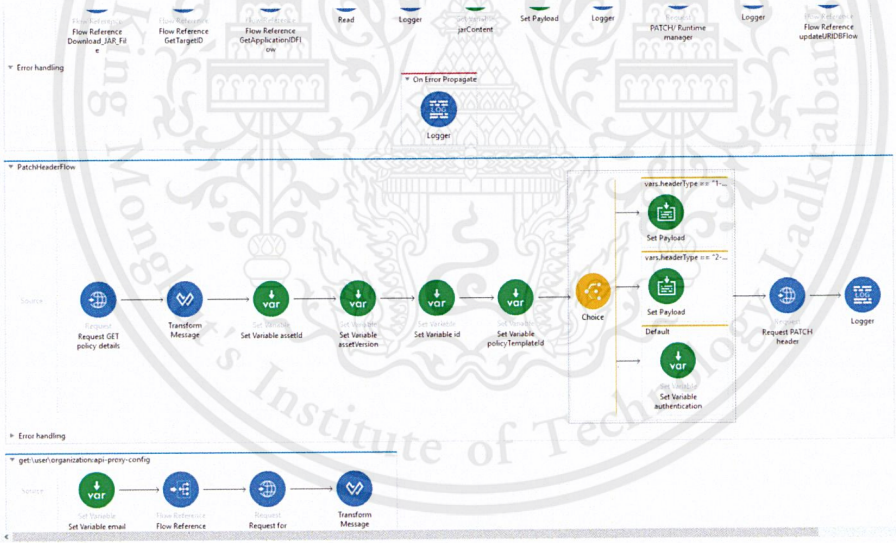

Figure 3.29 PatchHeaderFlow flow

- Figure 3.29 outlines PatchHeaderFlow, which is a flow to patch and change values of a Client ID and Secret for their API Proxy.
- **POSTheaderInjectionFlow flow:** The same flow as portrayed in Figure 3.23, to create an API Proxy process. To add Client ID and Secret to API Proxy that does not have any authentication.

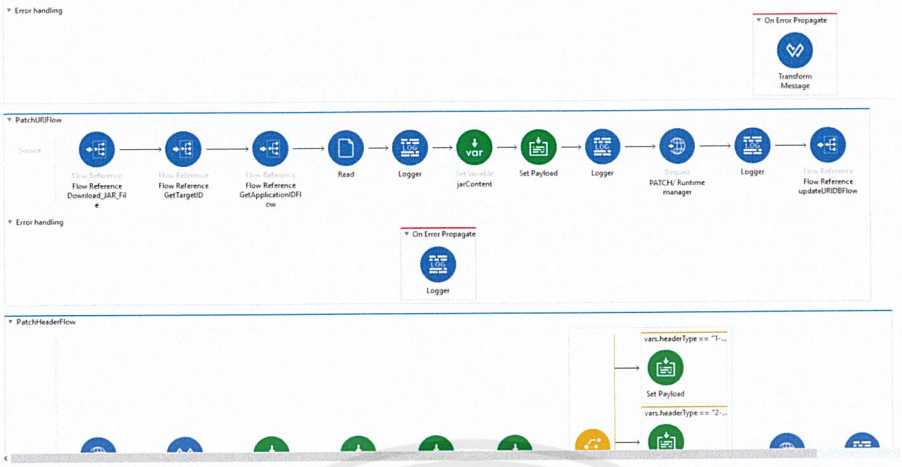

Figure 3.30 PatchURIFlow flow

- As outlined in Figure 3.30, PatchURIFlow flow is to change or update the URL of the API that the API Proxy protects or sends requests to.

3. Endpoint with POST method to promote API Proxy from an environment to another environment:

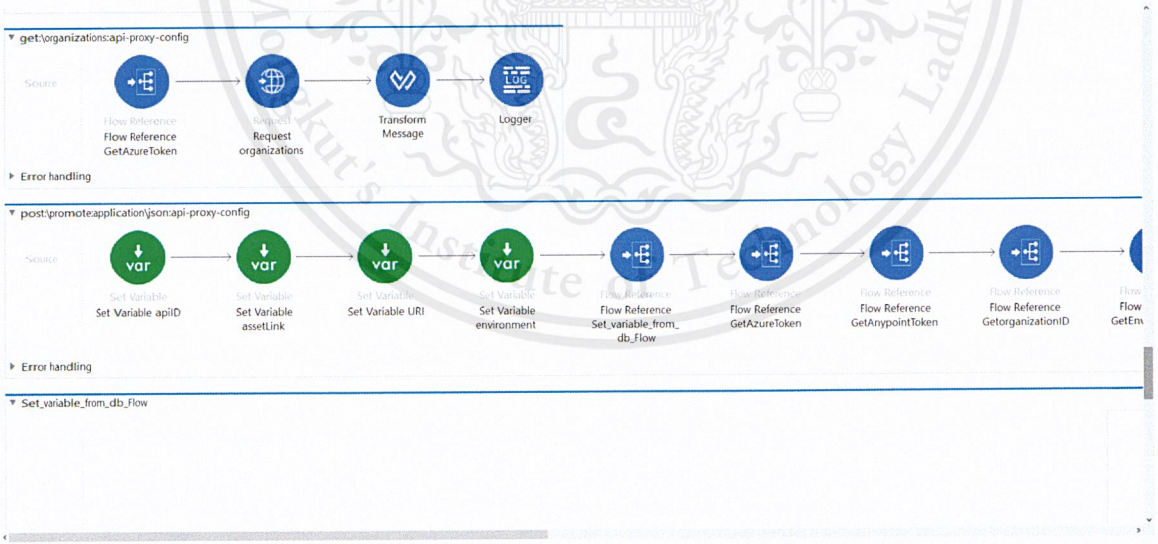

Figure 3.31 post:\promote:application\json:api-proxy-config flow

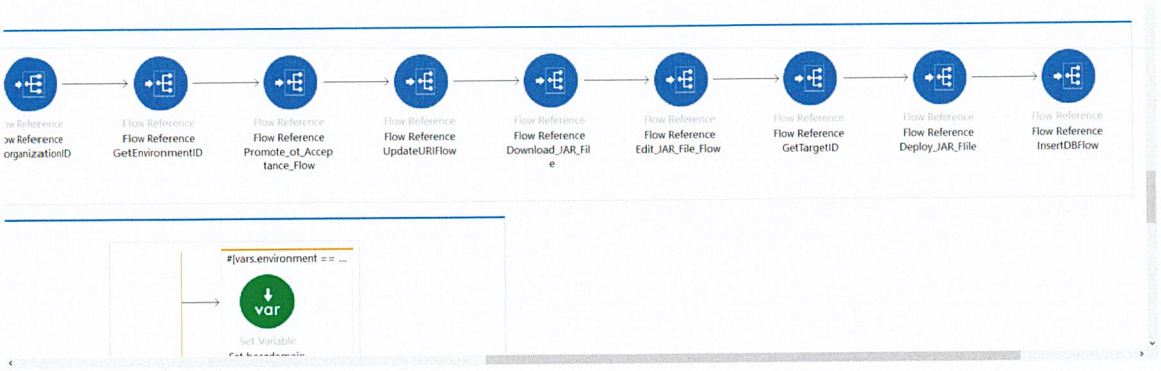

Figure 3.32 post:\promote:application\json:api-proxy-config flow

- Figures 3.31 and 3.32 shows the post:\promote:application\json:api-proxy-config flow. This is a flow that receives POST with JSON body that contains API ID, URL for API (in case the URL of API is different in each environment) and the environment that the user wants to promote the API to. All of the details inside this flow are similar to the process used to create API Proxy, except for some data that is received from the API Proxy that the user promoted in the database, which is used as a reference API Proxy.

4. Endpoint with DELETE method to delete API Proxy from Anypoint Exchange, API Manager, Runtime Manager and the database.

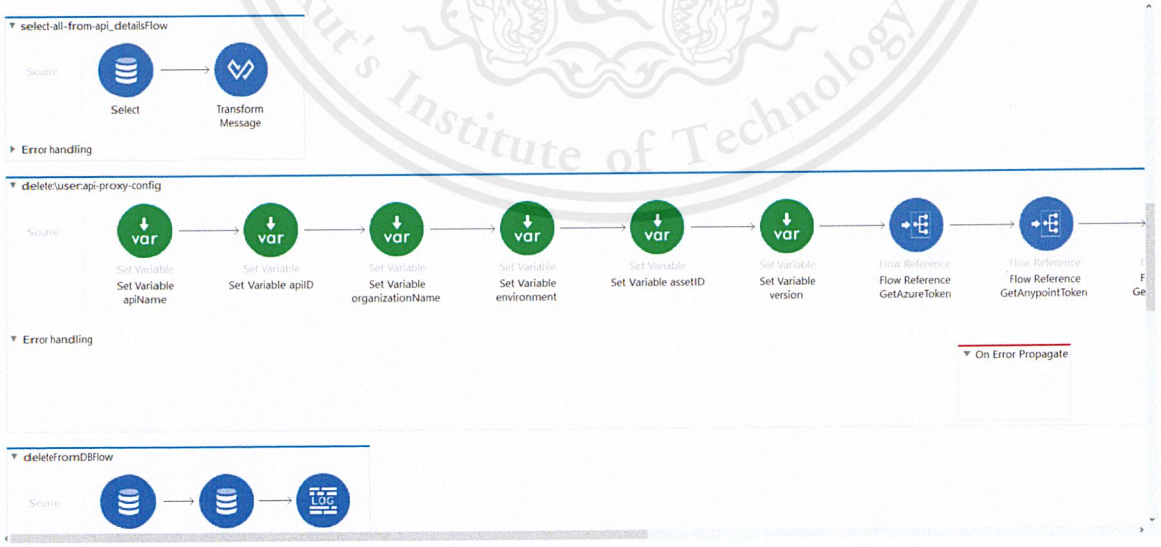

Figure 3.33 delete:\user:api-proxy-config flow

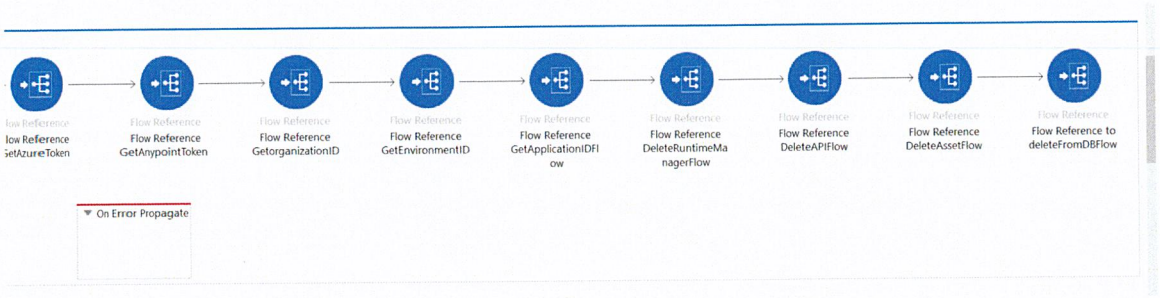

Figure 3.34 delete:\user:api-proxy-config flow

- As shown in Figures 3.33 and 3.34, delete:\user:api-proxy-config flow is a flow that receives the DELETE method from the request then references other flows to delete API Proxy from Anypoint Exchange, API Manager, Runtime Manager, and the database.

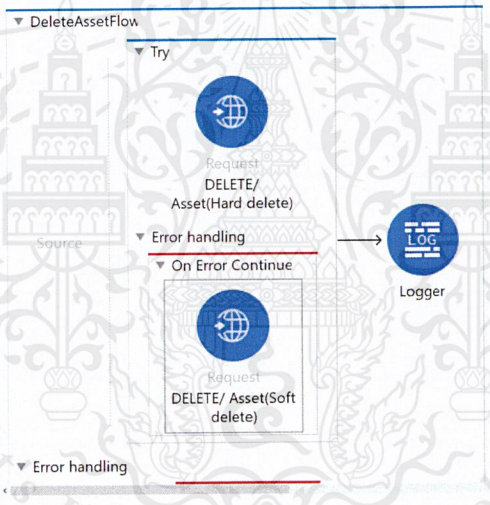

Figure 3.35 DeleteAssetFlow flow

- As demonstrated in Figure 3.35, DeleteAssetFlow flow is a flow to delete API Proxy from Anypoint Exchange.

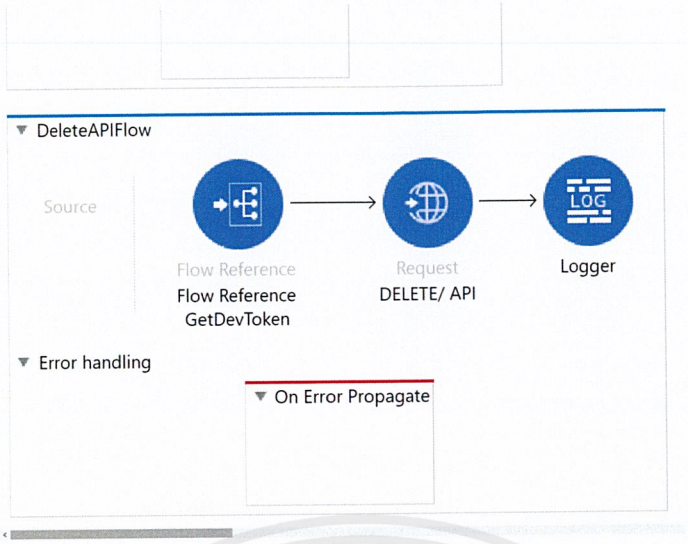

Figure 3.36 DeleteAPIFlow flow

- As outlined in Figure 3.36, DeleteAPIFlow flow is a flow to delete API Proxy from API Manager.

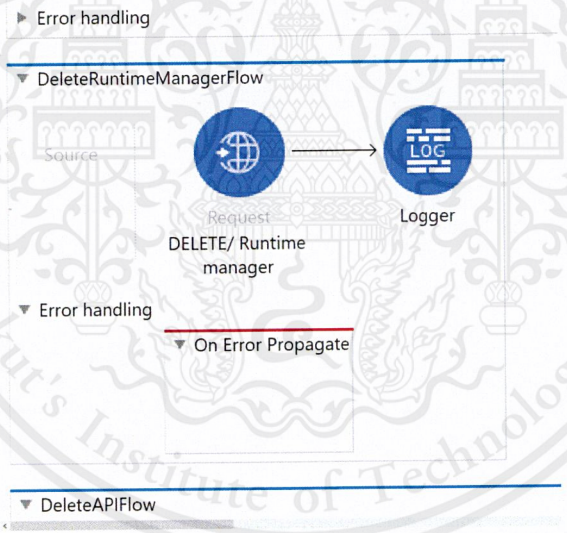

Figure 3.37 DeleteRuntimeManagerFlow flow

- As portrayed in Figure 3.37, DeleteRuntimeManagerFlow flow is a flow to Delete API Proxy from Runtime Manager.

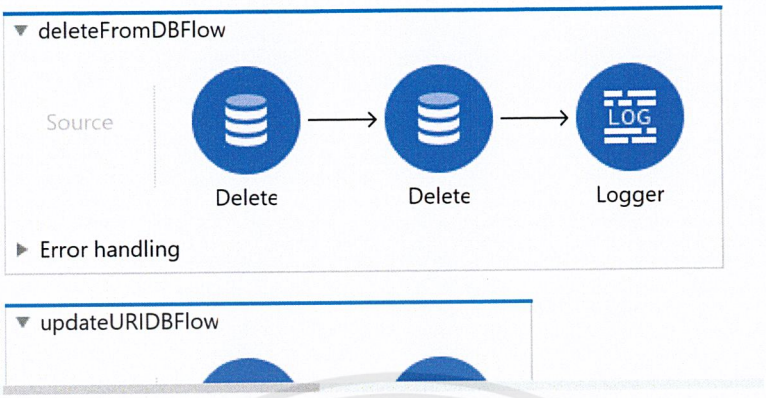

Figure 3.38 deleteFromDBFlow flow

– As demonstrated in Figure 3.38, deleteFromDBFlow flow is a flow to delete API Proxy from the database.

5. Endpoint with the GET method to get users who are currently using the EPROXC application in order to identify by client credentials inside an access token. This endpoint is called /me endpoint from XOM Mule API, which also requires client credentials from access token to get current user information. However, normally, when the application requests an access token from XOM Mule API, the credential that we receive is a credential of the application, so then we need to use On-Behalf-Of flow to keep the client credential of the user. Refer to Figure 3.39 to see what the On-Behalf-Of flow process looks like.

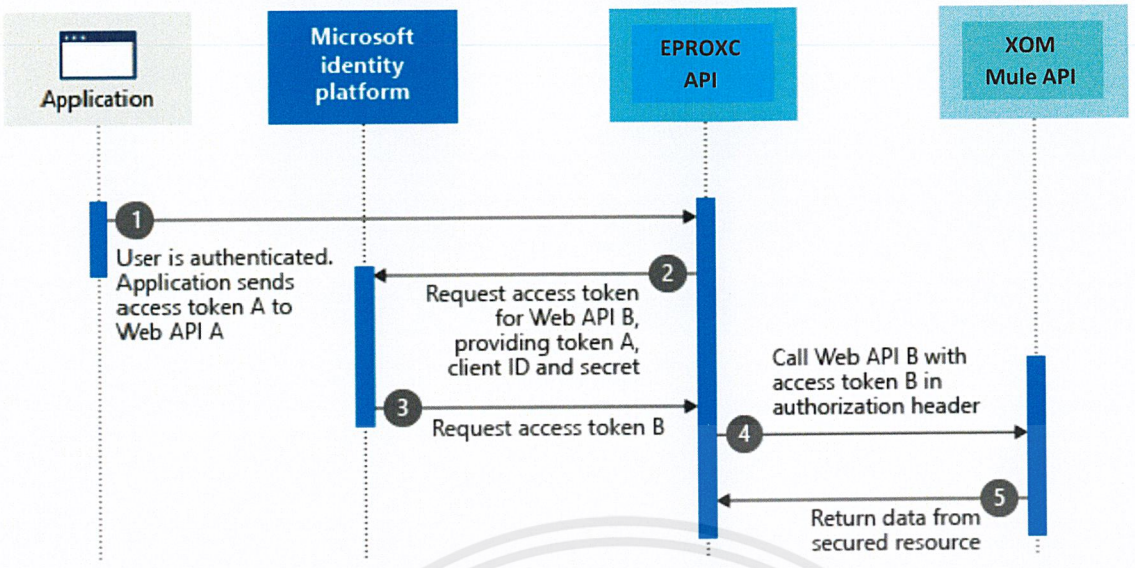

Figure 3.39 Example of On-Behalf-Of flow usage between EPROXC API and XOM Mule API

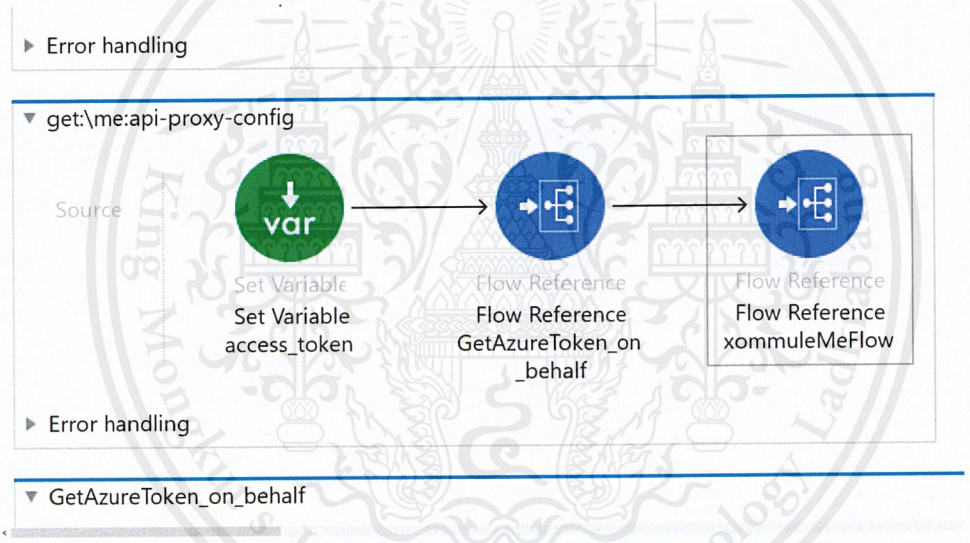

Figure 3.40 get:\me:api-proxy-config flow

- As shown in Figure 3.40, get:\me:api-proxy-config flow is a flow that receives GET request with access token on the query parameter. Subsequently, it references other flows to get user information from the access token.

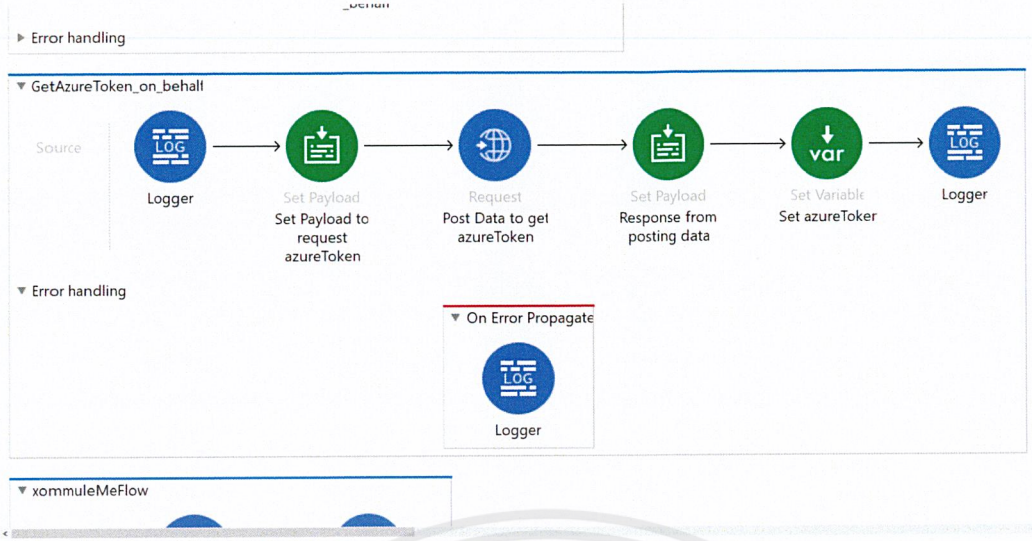

Figure 3.41 GetAzureToken_on_behalf flow

- As portrayed in Figure 3.41, GetAzureToken_on_behalf flow is a flow to Request an access token for accessing XOM Mule API using On-Behalf-Of flow with a user access token in order to keep the client credentials of the user in the requested token.

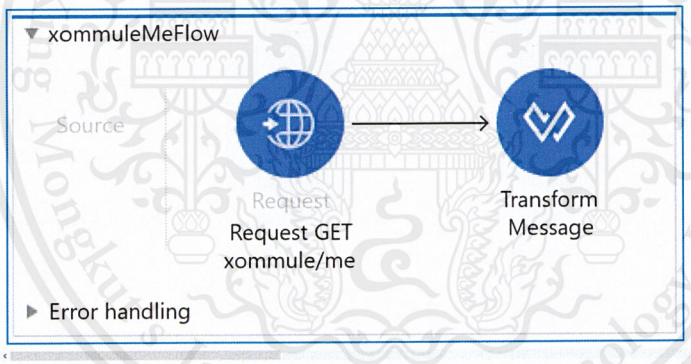

Figure 3.42 xommuleMeFlow flow

- As demonstrated in Figure 3.42, xommuleMeFlow flow is to request the user information from /me endpoint at XOM Mule API using the access token that was requested from **GetAzureToken_on_behalf flow**, as outlined in Figure 3.41, and then return user information.

3.2 EPROXC API deployment

To make the API work in the organization, API to OpenShift was deployed together with Runtime Manager, Asset and API Manager. Since the API was deployed to OpenShift, this means that the API can only be used for internal applications only.

3.3 Security of the API

EPROXC API has been protected by Azure Active Directory using the OAuth model, therefore, when a user wants to access this API, they need to provide an access token. Moreover, the default setting in Azure Active Directory is set so that only users in ExxonMobil can request for an access token for this API. Refer to Figure 3.38 for more information.

3.4 EPROXC Website

I also created a website using React.js framework with the purpose of making it easier for the user to create an API Proxy. The EPROXC website consists of 2 pages:

- **Home page:** This page will provide a form for the user to fill in necessary information in order to create an API proxy. However, before the user can access this website they need to pass the OAuth process, therefore, when users go to the website it will automatically redirect them to https://login.microsoftonline.com for the sign-in process, which will authorize the user and request for access token to access EPROXC API. The user will need an ExxonMobil account to sign in. Once the system has granted authorization to the user, they will get an access token to access the EPROXC API.

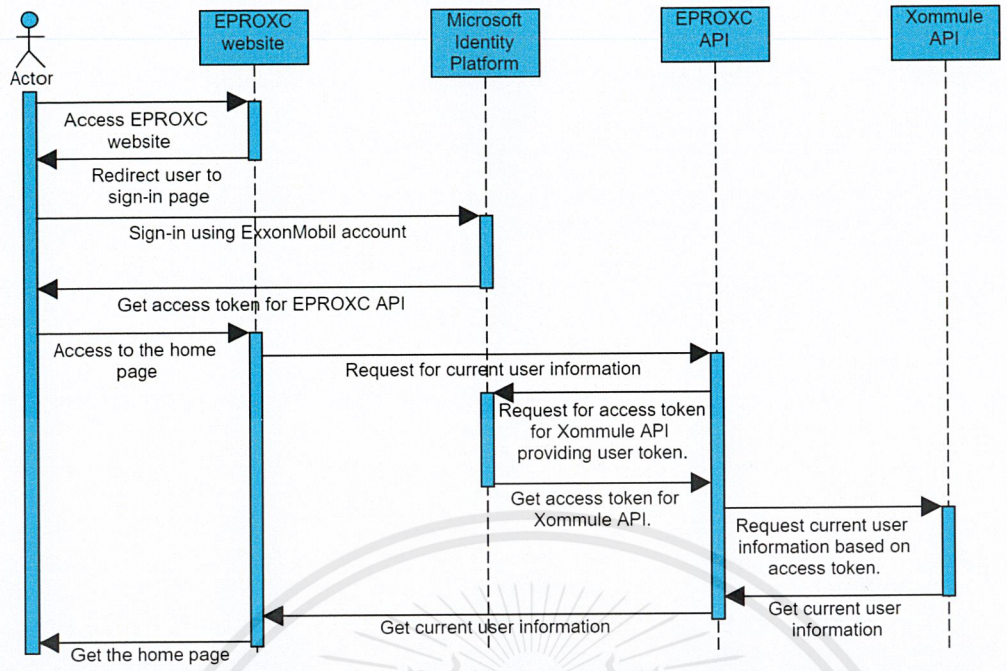

Figure 3.43 Sequence diagram of the home page

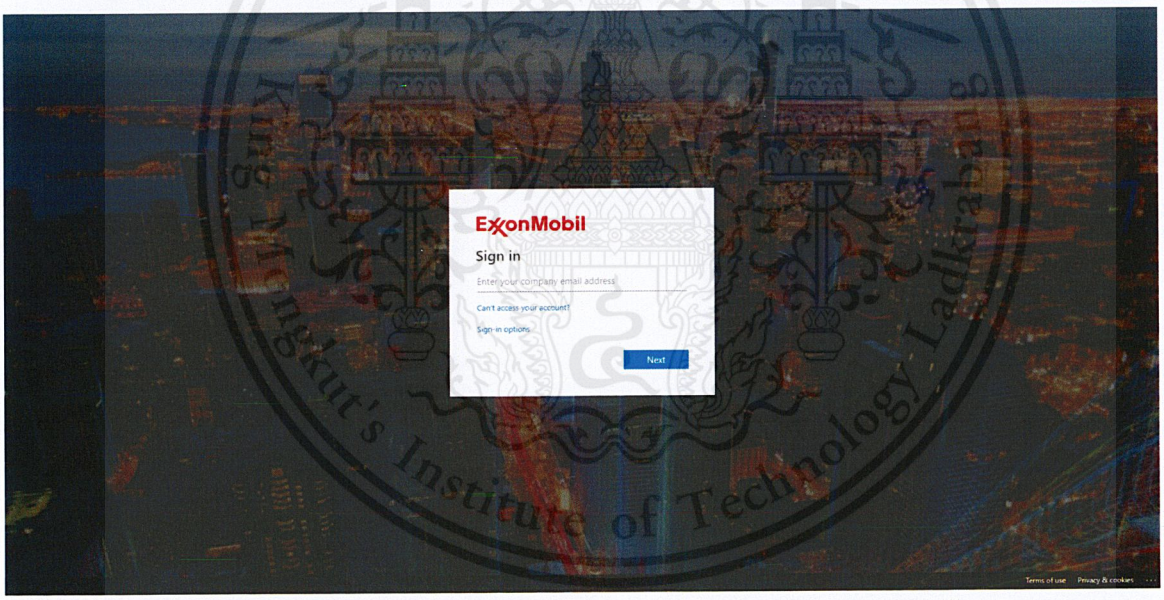

Figure 3.44 Example sign-in page

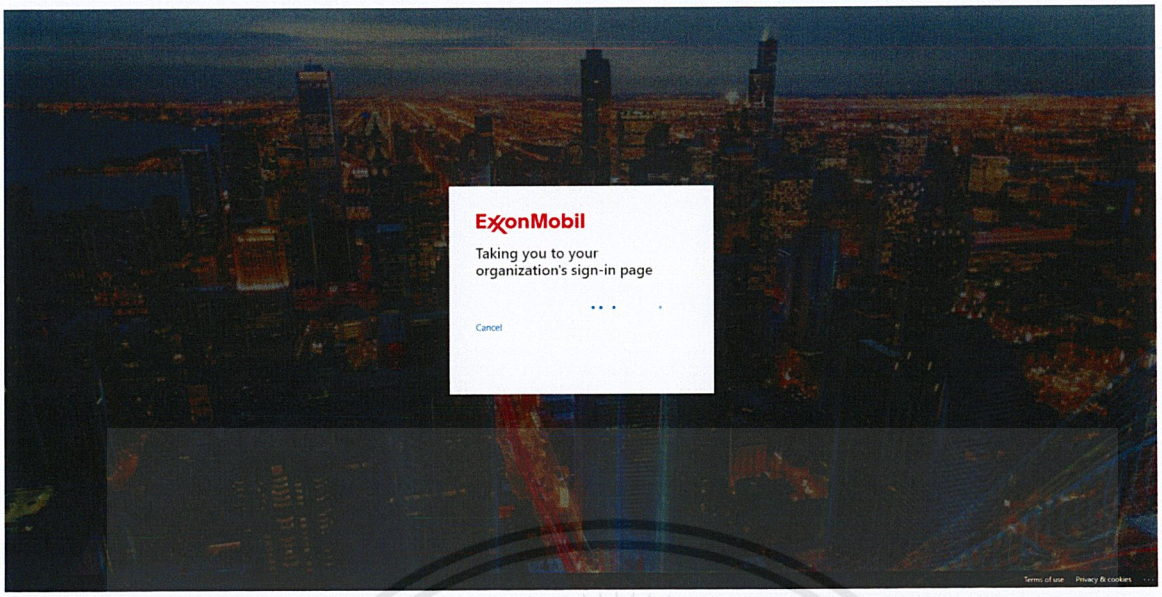

Figure 3.45 Example sign-in page

After signing in, the user will be redirected to the Home page, as shown in Figure 3.45. To create an API Proxy, the user must fill in all the required fields, as demonstrated in Figure 3.46, which are:

Figure 3.46 EPROXC website home page

- **API name:** Name of the API Proxy that the user wants to create.
- **API description:** Description of the API that the user wants to create. This description will appear on the asset in Anypoint Exchange.

- **API overview:** Overview of the API. This overview will also appear on the asset in Anypoint Exchange.
- **How the API work:** To tell other users how this API works on the asset in Anypoint Exchange.
- **Creator:** This field is fixed by the name of the current user based on the user information received from EPROXC API.
- **URL:** Base URL of the API that the user wants for their API Proxy.
- **Organization:** Choose an organization to deploy the API Proxy. All of the organizations are from user information received from EPROXC API, which are the organizations that the user has permission to create an API Proxy for in MuleSoft. If the user does not have permission, a button will pop up allowing the user to email our team in order to request permission to create API Proxy in the organization where they want to deploy the API Proxy.
- **Environment:** Choose an environment to deploy the API Proxy.
- **Deployment server:** Choose a server to deploy the API Proxy, between OpenShift or Cloudhub.
- **Authentication:** Choose authentication type for the API Proxy, from the following:
    - No authentication
    - Basic authentication
    - Headers injection (Header authentication)

After filling in all the required fields, the user must click submit and wait for the creation of the API Proxy process from EPROXC API to be finished. The user will be redirected to a page that shows all of the details of the API Proxy that was created.

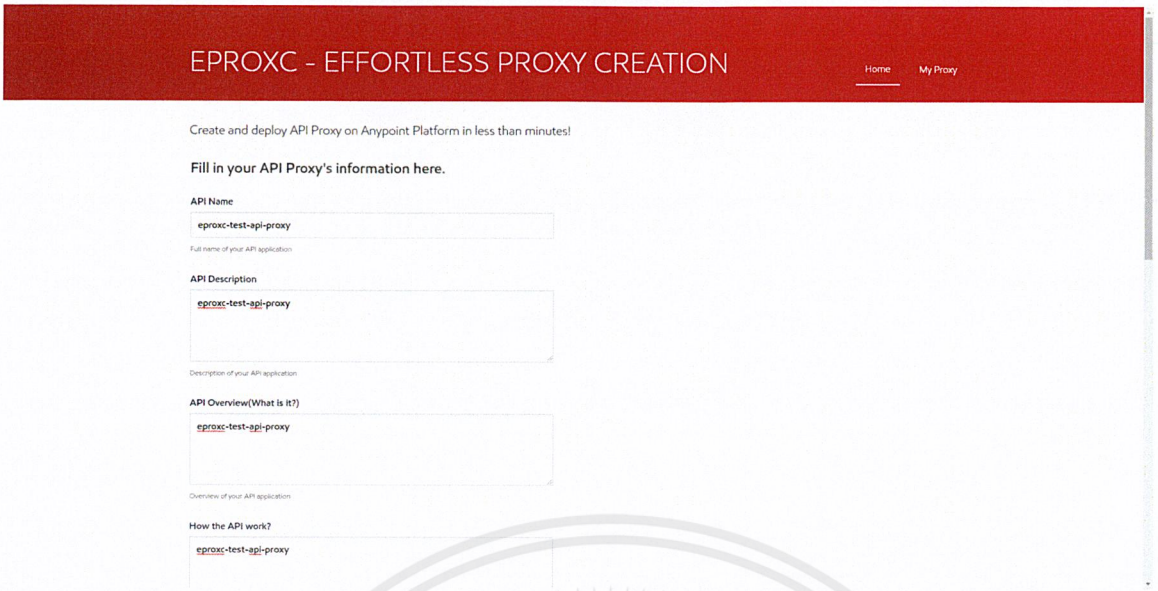

Figure 3.47 EPROXC website home page

- **My Proxy page:** My Proxy page looks like the images displayed in Figures 3.49 to 3.53. This is a page where the user can take a look at the information and status of the API Proxies they created and also at the API Proxies created in their organizations. Moreover, the user can go to API manager or Asset in Anypoint Exchange of their API Proxy and if they are the owner of the API Proxy, they can Change the URL of API, Add or change Client ID and Secret, Delete API Proxy (which means it will be deleted from Anypoint Exchange), API Manager, Runtime Manager and database, or the user can promote the API to another environment. To learn how to access the My Proxy page, refer to the diagram outlined in Figure 3.47.

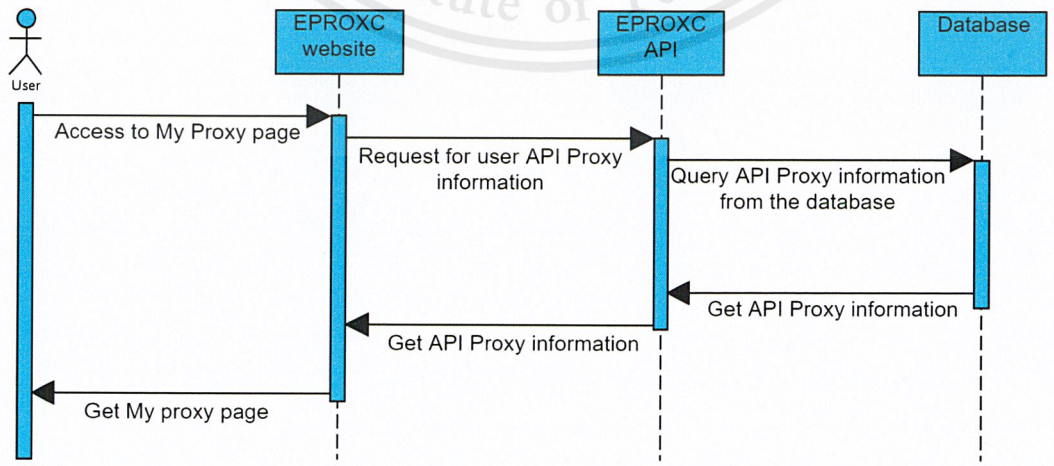

Figure 3.48 Sequence diagram of My proxy page

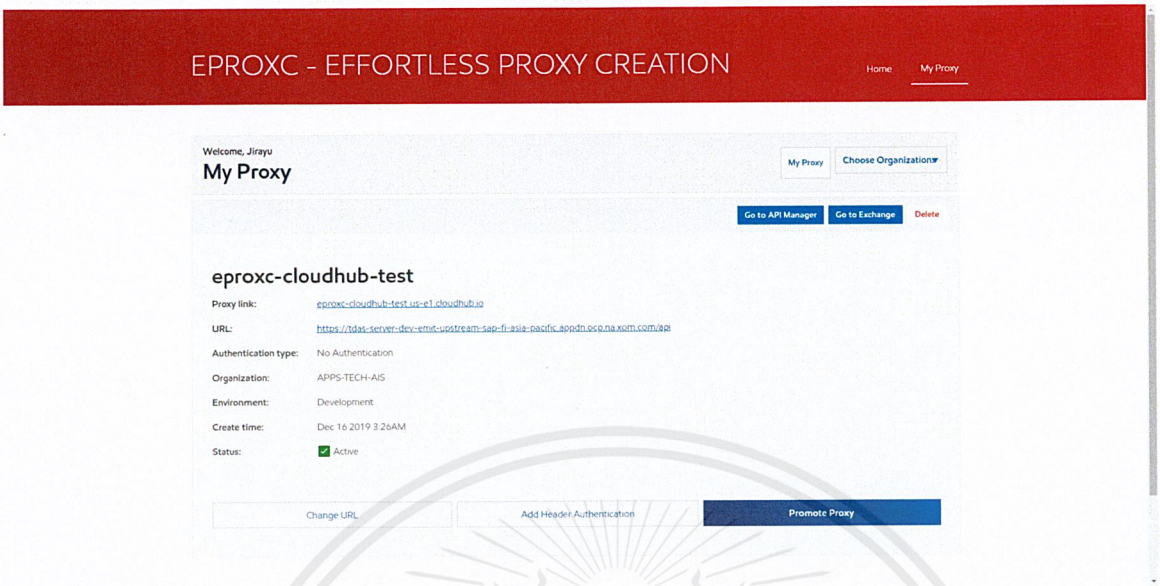

Figure 3.49 EPROXC website My Proxy page

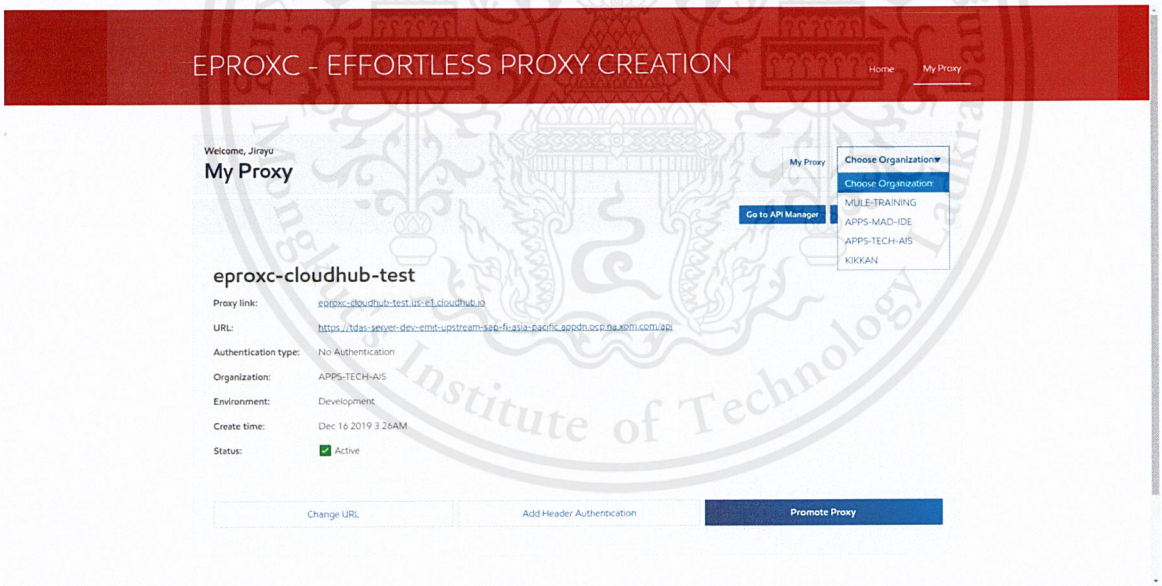

Figure 3.50 EPROXC website My Proxy page

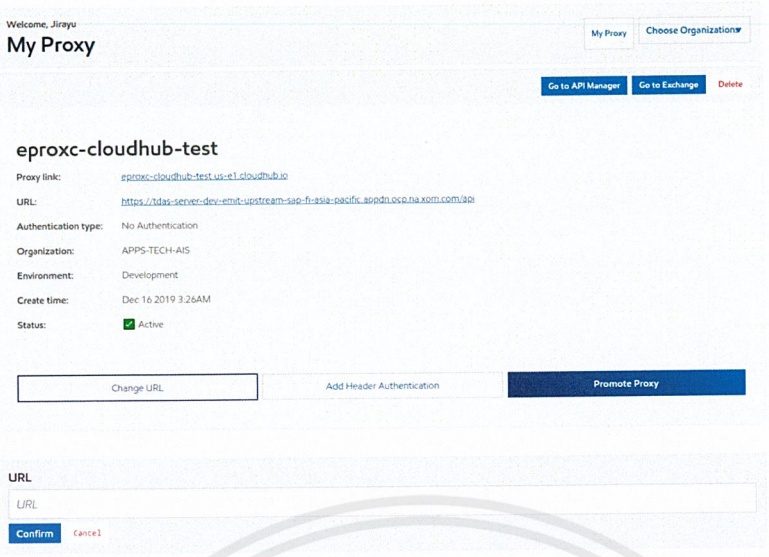

Figure 3.51 Example of changing API URL

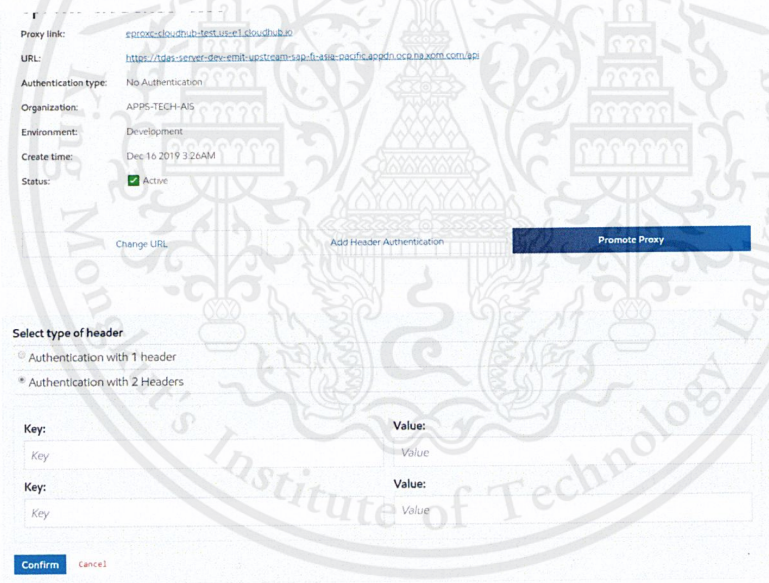

Figure 3.52 Example add or change Client ID and Secret

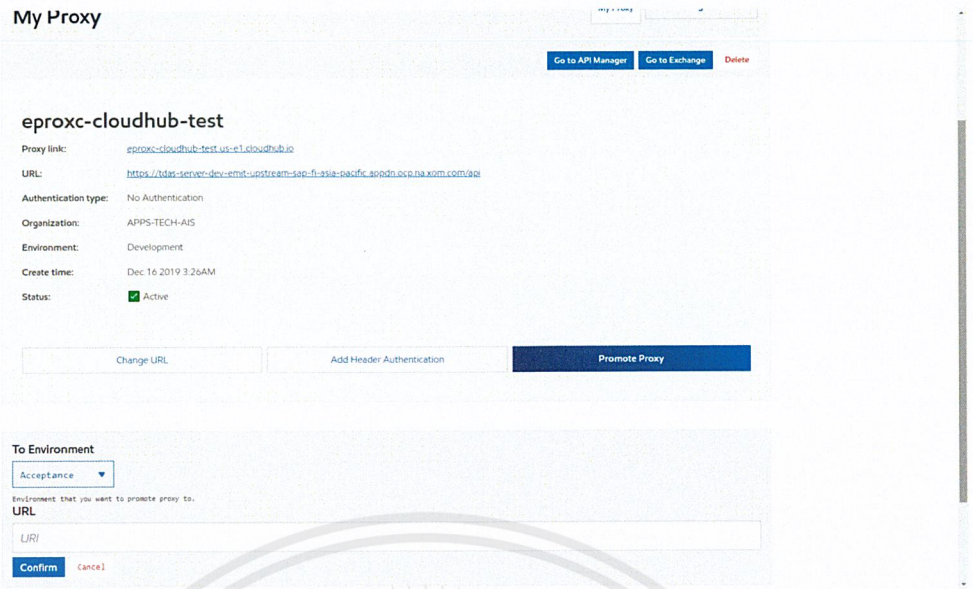

Figure 3.53 Example of promoting API Proxy to another environment

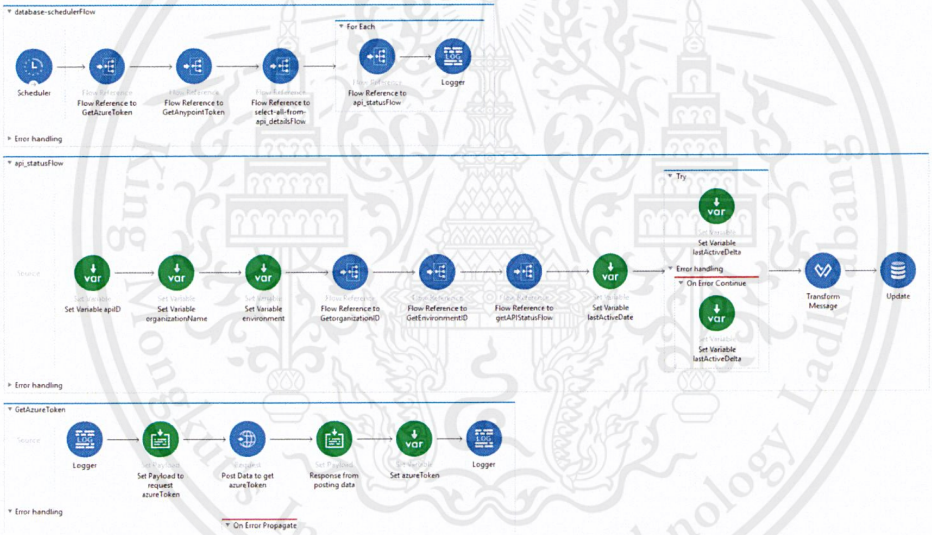

Figure 3.54 flows of EPROXC scheduler

Figure 3.55 flows of EPROXC scheduler

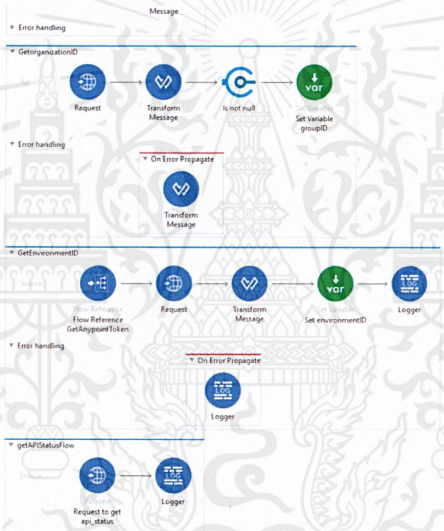

Figure 3.56 flows of EPROXC scheduler

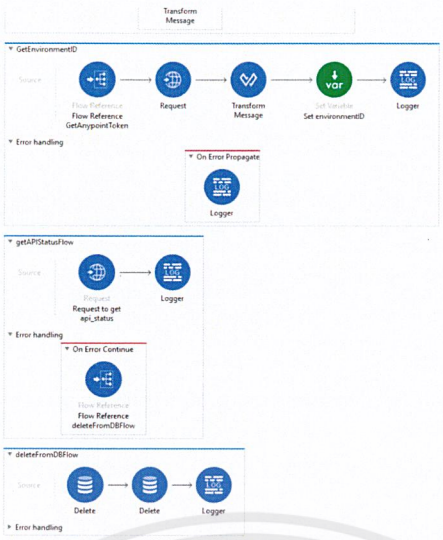

Figure 3.57 flows of EPROXC scheduler

3.5 EPROXC scheduler

Figures 3.54 to 3.57 show the process in the flow for scheduling a status update of all the API Proxies created by EPROXC. Here, a schedule was created to check the status of the API Manager from Anypoint Platform of the API Proxy and then to update status in the database.

3.6 Database

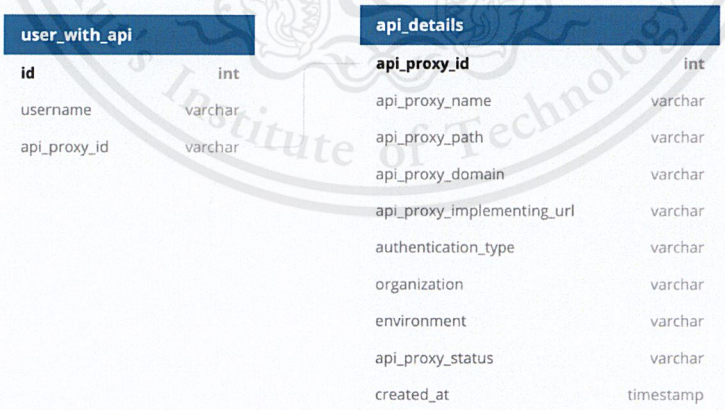

Figure 5.8 Database diagram of EPROXC

Chapter 4

Result

This chapter will discuss the result of the project. What are the end products and what are the benefits of those products?

4.1 EPROXC website and API

Users can access the EPROXC website, as shown in Figure 3.45, to easily create an API Proxy or they can go to Anypoint Exchange and search for EPROXC API to use in their application. After creating the API Proxy using EPROXC, the user will be able to use 3 products from MuleSoft, which are, the asset on Anypoint Exchange, API manager, where they can set policies for API Proxy, and Runtime Manager to deploy running and monitoring of the API Proxy. Other benefits of the EPROXC are:

4.1.1 Saving time and effort for the creation of API Proxy

Before EPROXC, the old process required a lot of effort in order to create one API Proxy for the API, especially for developers who never used a MuleSoft product before, because they needed to learn everything about MuleSoft in order to create API Proxy. EPROXC reduces a lot of effort and complexity in the creation of an API Proxy because one can create an API Proxy on Anypoint Platform without having any knowledge of MuleSoft. Rather than working with the old process of creating API Proxy, which involves many processes and takes a lot of effort, now the users can use EPROXC and just fill in some required information then wait for API Proxy to be created.

4.1.2 External application can request to internal API

Because EPROXC allows a user to deploy API Proxy to Cloudhub, if the user uses this API Proxy with internal API it will allow an external application to call to the internal API by requesting the API Proxy, instead of requesting the internal API directly.

4.1.3 Secure API

By using an API Proxy, one can set policies to secure their API, for example, Basic authentication, JWT token validation for validating access token in OAuth process,

Rate limiting to limit request from users, etc. EPROXC brings more security to the API not only because of policies but due to the fact that the URL is the URL of the API Proxy, which means the user will not call the API directly but will instead access the API Proxy.

4.1.4 Centralize API in Anypoint Exchange

Even if a user creates their API without using MuleSoft, they will be able to have an asset with a template that tells other users about the documentation of their API on Anypoint Exchange. Furthermore, if everyone in ExxonMobil uses EPROXC, this would mean that we can search all APIs on Anypoint Exchange, which is the best scenario for developers to develop their application because when they need an API they can just search it from Anypoint Exchange.

4.1.5 MuleSoft Products

Users can use Anypoint Exchange, API Manager and Runtime Manager to improve their API. Additionally, they can monitor and manage their API using those products.

4.1.6 Easy to access, manage and delete API Proxy

Based on the old process, once the API Proxy has been created in OpenShift, the user will not know the URL of the API Proxy. However, with EPOXC, the user can view the information of their API Proxy from the website and they can easily edit, go to API Manager, go to asset on Anypoint Exchange, edit some configurations and delete API Proxy on EPROXC website. They can also check the status of their API.

4.1.7 Monitoring and health check API Proxy

Runtime Manager allows the user to monitor and check the health of the API Proxy that has been created from EPROXC by using /healthz path on the API Proxy URL. Therefore, the format for the health check path will be **"{Domain}/apps-tech-ais/common4/{API Proxy name}/v1/api/healthz"** and if it is working, the response message will be OK with response status 200 OK. It should be noted that this health check path only works for API Proxy that deploys to OpenShift

Chapter 5

Summary

To conclude, EPROXC is a tool which includes 2 platforms, website and API, that automatically creates an API Proxy together with MuleSoft product asset on Anypoint Exchange, API Manager and Runtime Manager. EPROXC also provides a website that a user can easily create an API Proxy by filling in some required information on the website and displays a page where the user can see information, check the status of API Proxy, that they created, and also the API Proxy in their organizations. However, they can only edit or delete the API Proxy that they created and own. There are two main target servers when deploying the API Proxy: OpenShift, which allows the use of an internal application with internal API, and Cloudhub, which allows the use of an external application with an internal API. After creating an API Proxy using EPROXC, the user will have an asset on Anypoint Exchange, which is an API Proxy documentation for other developers to use the API Proxy. Furthermore, the user will be able to set policies for their API Proxy in order to secure their API Proxy using API Manager. Moreover, the user can monitor their API Proxy using Runtime Manager and view the log to know what is going on when some error happens in their API Proxy. EPROXC has provided many benefits to the ExxonMobil organization, especially for API developers. Not only does it save time, when creating an API Proxy for their API, but also for securing, managing and monitoring the API. For developers who want to implement their application, the benefit of EPROXC is that everyone in ExxonMobil will use all of the API with the document, therefore, it will be centralized on Anypoint Exchange. This makes it a lot easier for developers to access the API that they want or even an API that they never think they had. This is very useful for every organization, not only in ExxonMobil, because it saves time on implementing the application.

References

[1] Robert Wallach. (November 27, 2019). What is an API Proxy?

Retrieved from https://stoplight.io/blog/api-proxy-vs-api-gateway-c008c942a02d/

[2] MuleSoft. (2019). What is an API?

Retrieved from https://www.mulesoft.com/resources/api/what-is-an-api

[3] Arun Dorairajan. (October 4, 2019). API Proxy diagram image

Retrieved from https://apifriends.com/api-security/api-proxy-vs-api-gateway/

[4] Roger A. Grimes and Josh Fruhlinger. (September 20, 2019). What is OAuth?

Retrieved from https://www.csoonline.com/article/3216404/what-is-oauth-how-the-open-authorization-framework-works.html

[5] Microsoft Azure. (2019). What is On-Behalf-Of flow?

Retrieved from https://docs.microsoft.com/en-us/azure/active-directory/develop/v2-oauth2-on-behalf-of-flow

[6] Mitchell Anicas. (July 21, 2014) OAuth diagram image

Retrieved from https://www.digitalocean.com/community/tutorials/an-introduction-to-oauth-2

[7] Mulesoft. (2019). Anypoint Platform website

Retrieved from https://anypoint.mulesoft.com/home/.

[8] ExxonMobil. (2019). Unity website image

Retrieved from http://unity.na.xom.com/

[9] Chris Stetson of NGINX, Inc. (August 15, 2018). What is Modern Application?

Retrieved from https://www.nginx.com/blog/principles-of-modern-application-development/

[10] Redhat. (2019). Modern Application example Image

Retrieved from https://image.slidesharecdn.com/modernapplicationdevelopment-ghv1-0-171204225228/95/modern-application-development-v10-14-638.jpg?cb=1512428040

[11] Margaret Rouse. (2015). What is MuleSoft?

Retrieved from: https://searchcloudcomputing.techtarget.com/definition/MuleSoft

Appendix

Appendix A

Modern Application:

A modern application is an application that supports multiple users with no barrier in UI. Whichever platform they are using, all of them will be connected to the application through an API. Modern application is designed for users on a large scale level.

A modern application provides an API or Application Programming Interface for users to access data and services. API is available via an internet connection over HTTP or HTTPS. All features and functionality are available through the GUI (Graphical User Interface) or CLI (Command-line Interface).

Data provided by API mostly comes in JSON format to represent the objects and services.

Modern applications are built on top of a modern stack to help developers create application API easily. It is easy for the application to consume and monitor JSON data.

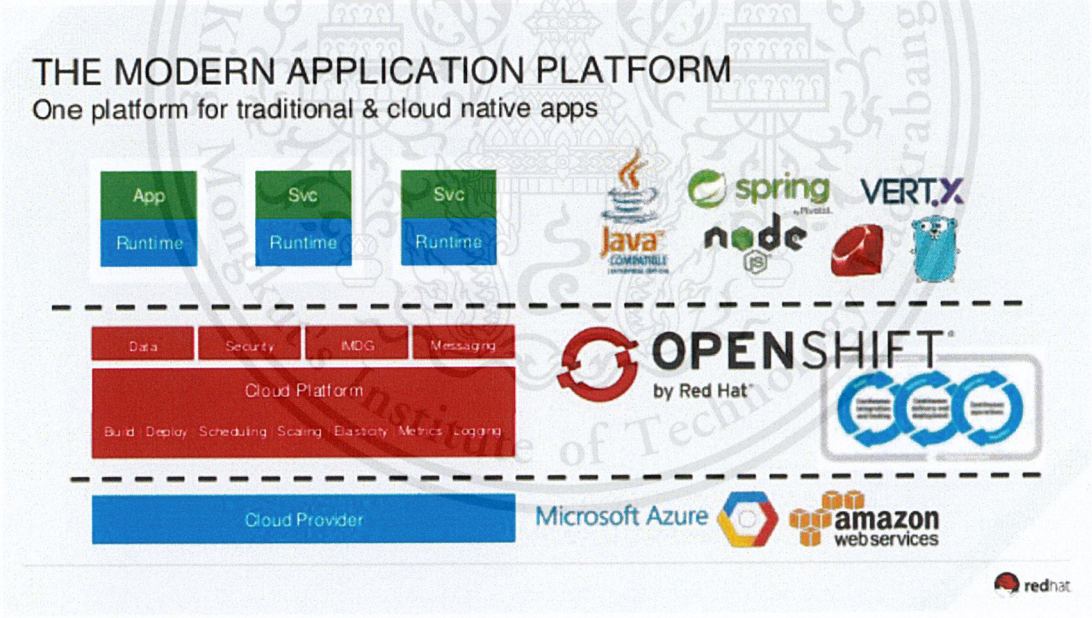

Figure A.1 Example of Modern Application

MuleSoft:

MuleSoft is a product that provides almost everything that the API needs.

Main components of MuleSoft:

MuleSoft's Anypoint Platform, shown in Figure A.2, offers a number of tools and services, including the following:

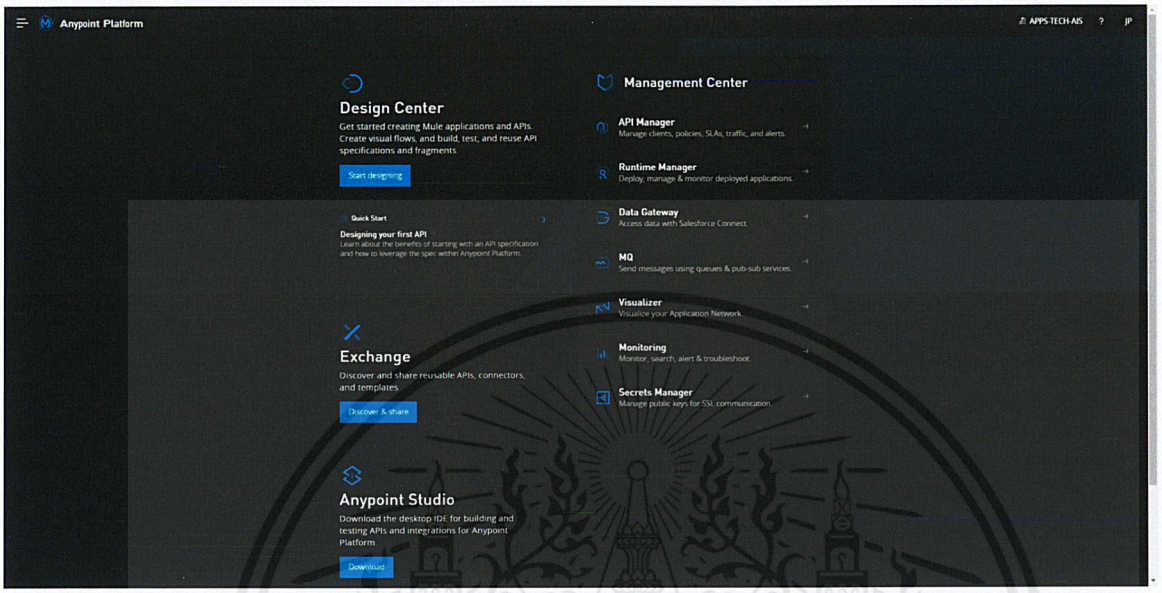

Figure A.2 Example page of Anypoint Platform

- <u>API Manager</u>

An interface for developers to manage APIs, as well as secure them via an API gateway or API Proxy. By using the API Manager, it is possible to control access of users to the APIs, secure connections between clients and back-end data sources and create policies for advance security on APIs around API calls and throttling.

- <u>Anypoint Studio</u>

A graphical, Java-based design environment that a developer can use to develop and test their API in advance using a local environment. Furthermore, they can deploy APIs to on-premise and cloud environments. Anypoint Studio also includes many features that allow users to map, build, edit and debug data integrations.

- Anypoint Runtime Manager

A central console for users to deploy, run and monitor application.

- Anypoint Exchange

A place where all APIs are in one place, which offers efficiency for developers to use it to store and access APIs, templates, documentation and other resources.

- Anypoint Monitoring

A dashboard to monitoring application and for application health check.

- CloudHub

A multi-tenant integration platform as a service (iPaaS), or a server for running the API like OpenShift. CloudHub offers a management service, which means a development team does not need to install or operate any hardware or software in order to use it.

Biography

Cooperative Title: EPROXC-Effortless Proxy Creation

Name: Jirayu Promsongwong

Student Number: 59011599

Faculty: Engineering

Department: Computer Engineering

Major: Computer Innovation Engineering

Personal Information

Date of Birth: 10 January 1998

Phone: 0959026252

Email: bossnohot1@hotmail.com

Education

- Bachelor's degree in Computer Innovation Engineering at King Mongkut's Institute of Technology Ladkrabang.