

**Salesforce Marketing Cloud Integration Architecture**

**Document**



**Version: 1.0**

**Author: Suraj Desai**

**Date: 09-September-2020**

Version: 1.0

Date: 31/12/2019

Author: Suraj Desai

Table of Contents

[Revision Log 2](#_Toc51776489)

[Reviewers 2](#_Toc51776490)

[Approvers 2](#_Toc51776491)

[Glossary 2](#_Toc51776492)

[Purpose of this Document 3](#_Toc51776493)

[Overview 3](#_Toc51776494)

[Key drivers for the solution. 4](#_Toc51776495)

[Integration Architecture 4](#_Toc51776496)

[Dell Boomi (iPAAS) 8](#_Toc51776497)

[Monitoring 8](#_Toc51776498)

[Failover 8](#_Toc51776499)

[Logging 8](#_Toc51776500)

[Non-Functional Requirements 9](#_Toc51776501)

[Assumptions 9](#_Toc51776502)

[Dependencies 9](#_Toc51776503)

[Open Questions 9](#_Toc51776504)

[Appendix Section 9](#_Toc51776505)

# Revision Log

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Author | Version | Changes |
| 09-September-2020 | Suraj Desai | 1.0 | Initial Draft |
|  |  |  |  |

# Reviewers

|  |  |  |
| --- | --- | --- |
| Date | Reviewer | Title |
|  | Sylvain Ottle |  |
|  | Kunal Doshi |  |
|  |  |  |

# Approvers

|  |  |  |
| --- | --- | --- |
| Date | Approver | Title |
|  | Rajeev Venkitaraman |  |
|  |  |  |
|  |  |  |

# Glossary

|  |  |
| --- | --- |
| Term | Description |
| API | Application Programing Interface is a computing interface which defines interactions between multiple software intermediaries. It defines the kinds of calls or requests that can be made, how to make them, the data formats that should be used, the conventions to follow, etc. |
| Integration | Integration is the process of determining and building the relationship of information across systems or applications. |
| REST | Representational state transfer is a software architectural style that defines a set of constraints to be used for creating Web services. Web services that conform to the REST architectural style, called RESTful Web services, provide interoperability between computer systems on the internet. |
| iPAAS | Integration Platform As A Service is a cloud: Cloud-based integration is a form of systems integration business delivered as a cloud computing service that addresses data, process, service-oriented architecture and application integration. |
| Dell Boomi | Dell Boomi is the iPaas platform that is provisioned at Michelin for hybrid and cloud to cloud integration use cases. |
| SAAS | Software as a service is a software licensing and delivery model in which software is licensed on a subscription basis and is centrally hosted. |
| Marketing Cloud | Salesforce Marketing Cloud is a SAAS platform that enables business to speak to customers with the right message, at the right time, with intelligent marketing automation. |
| Apostrophe | Is a Content management system which is used at Michelin to develop and host B2B some B2B website for Michelin and its brands. |
| B2B | Business-to-business is a situation where one business makes a commercial transaction with another Business. |
| B2C | Direct-to-consumer refers to selling products directly to customers, bypassing any third-party retailers, wholesalers, or any other middlemen. |

# Purpose of this Document

This document provides a high-level integration architecture and solution for integrating various Salesforce clouds with other systems at Michelin. As part of this integration architecture we will also cover Michelin Customer API (both B2B and B2C) that will give single view of the Customer across all the 3 clouds. The intended audience for this document is Architects and Stakeholders from Engage, API and Apostrophe teams.

This document describes the integration architecture and methods to put in practice for all integrations within Salesforce scope, in and out. API led approach is has been chosen as primary approach in creating the integrations across the different SF clouds, which provides uniform, quick and controlled access to data across SF clouds via APIs and helps to rapidly built reusable business use cases on top of these data APIs with speed and agility.

# Overview

Salesforce Marketing Cloud is being provisioned at Michelin as the digital marketing platform for B2C and B2B Campaigns communication via email and SMS, marketing automations and lead generation through Cloudpages.

As a first step, the SFMC needs to be integrated with Apostrophe, which is a content management system that hosts Michelin B2C and B2B (for some countries) websites. There are also Promotion houses that send promotion information as a batch feed to SFMC.

Following is the context diagram systems involved in the SFMC integration.



*Figure 1: Context Diagram Marketing Cloud Integration*

## Key drivers for the solution.

1. Direct integration between systems and SFMC is to be avoided to prevent the problem of spaghetti integration.
2. The solution developed must be able to support future integration needs to SFMC without having to create new ingestion points for every new system. Hence the solution must promote reuse.
3. As SFMC needs to support both B2C and B2B data models, validating incoming data against different models must be supported to ensure that the right data is ingested according to the Customer type.
4. The quality of the data that is being ingested into SFMC must be maintained by have a validation in place.
5. The access to the SFMC need to be controlled and managed efficiently.
6. Relevant security policies need to be in place so that only approved users/system can ingest or retrieve data from SFMC.
7. The solution must support data transformation, if any system is not able to send the data in the format required by SFMC.

# Integration Architecture

Keeping in mind the key driving factors an Integration architecture needs to be in place that will not only solve the current integration challenge but will also form a reusable base or all future integrations with Marketing Cloud.

An API led approach to integration will be used to build the overall integration or connectivity. Please see the **Conceptual Architecture view** and **API-led connectivity** in the [Appendix](#_Appendix_Section) section for more information on the API led connectivity.

The diagram below shows the High-Level Integration architecture that will be implemented for the Marketing Cloud integration with various user interaction layers like B2B and B2C websites, Mobile Apps, Concare call center app, etc.



*Figure 2: API-Led Integration Architecture for Marketing Cloud Integration*

As shown in the figure 2, the API-Led connectivity will be established using 2 layers of APIs. The Top layer, also known as the Experience API layer, will encompass APIs that are built for specific purpose. These are built for a specific business or technical use case. Example for such APIs are the Newsletter Subscription API, Contact Us API, Survey API and Promotion API.

All the Experience APIs are derived or built out from a single API, which is the Marketing Customer API. This layer of APIs is known as System APIs as they provide connectivity to an underlying system of record, which in this case is the Marketing Cloud. This API is built for reuse, easy and quick access to the Customer data stored in the Marketing Cloud. This also provides a layer of controlled access and data validation of the Marketing Cloud Customer Data. Team and Business units can built various APIs and Apps using this API to suit their business requirements.

Let’s look at how use case for Newsletter subscription API is built in an end to end manner using the API Led Approach



*Figure 3: Newsletter API using the Marketing Customer API*

As shown in the Figure 3, a user of a Michelin B2C website navigates to the form where he/she fills out contact information and also their preference for any newsletter subscription. When the form is submitted the following actions occur in order as described below

1. The form that the submitted on the website hosted on Apostrophe CMS, calls the Newsletter subscription API that is hosted on the Michelin iPaas (Dell Boomi). The API is exposed via the Enterprise APIM solution with provides security and policy enforcement features.
2. The Newsletter API then calls the Marketing Cloud Customer API. The Customer Data is validated against a B2B or a B2C schema depending on the Customer type. The Customer API then orchestrates between the internal out of box Marketing Cloud APIs to create a Customer Object in the Marketing cloud.
3. A success or a failure response is then communicated back to the calling Newsletter API, when in turn drives the appropriate page display to the user.

**Marketing Customer API**

This API will provide the ability to store and retrieve Customer information from the Marketing cloud. It will be a single point of entry to the Marketing cloud. It will provide data validation and access control to the data in and out of the Market Cloud.

Let’s see how this API and the overall API Led connectivity will help to achieve the key factors listed in the section [above](#_Key_drivers_for)

1. Direct integration between systems and SFMC is to be avoided to prevent the problem of spaghetti integration.

*-🡪 As there is no direct integration been Marketing Cloud and other systems, we will have in place an architecture that enables loose coupling. So, any changes in one system will not affect the other. Also, as the number of systems in the Enterprise grow, this will help us reduce the system to system integration points thereby avoiding the problem of spaghetti integration.*

1. The solution developed must be able to support future integration needs to SFMC without having to create new ingestion points for every new system. Hence the solution must promote reuse.

*-🡪 The Marketing Customer API will be a reusable component, which can be leveraged to build many other use cases that need Customer data from Marketing Cloud.*

1. As SFMC needs to support both B2C and B2B data models, validating incoming data against different models must be supported to ensure that the right data is ingested according to the Customer type.

*-🡪 The Marketing Customer API will have in place validation against a Customer type (B2B or B2C) that will enable Marketing Cloud to support 2 different data models without having to put in place any conditional validation the Marketing Cloud itself.*

1. The quality of the data that is being ingested into SFMC must be maintained by have a validation in place.

*-🡪 The data validation should help us maintain the quality of the data that is being ingested. Furthermore, if required, information such as email id and addresses can be validated and cleansed using third party APIs before the data ingestion into the Marketing Cloud*.

1. The access to the SFMC need to be controlled and managed efficiently.

*-🡪 Since the access in and out of the Marketing Cloud is via the Marketing Customer API, the access control can be managed more efficiently and at a single place by exposing the API on the Michelin APIM.*

1. Relevant security policies need to be in place so that only approved users/system can ingest or retrieve data from SFMC.

*-*🡪 *Since the access in and out of the Marketing Cloud is via the Marketing Customer API, the access control can be managed more efficiently and at a single place by exposing the API on the Michelin APIM.*

1. The solution must support data transformation, if any system is not able to send the data in the format required by SFMC.

-🡪 The Experience APIs that are purpose built will handle message transformation required to meet the format required by the Marketing Customer API.

# Dell Boomi (iPAAS)

As mentioned in the above section Dell Boomi will be used to build the Marketing Cloud Customer API. The Experience APIs can be build using the same iPaas platform or can also be built using any other relevant technology like Nodejs, Spring Boot and Azure depending on the capabilities of the teams that need to built the specific use cased using the Marketing Cloud Customer API.

Monitoring

Since this is a SAAS offering from Dell, overall, platform/infrastructure health and availability is monitored by Dell. You can check the status of the platform here: <https://status.boomi.com>. Updates on potential issues are sent to DIF (HIP/SPOC) team who will then notify all Michelin Boomi users about the same.

An application (API) specific alerting mechanism will have to be built while developing the API so that application (API) related errors are captured and intended stakeholders are notified on the same. Since the Marketing Customer AP is a synchronous REST API, any validation errors, or runtime errors are directly sent back to the consuming application as error response. Standard error HTTP codes can be used here for reporting the error back in the response.

Failover

Dell Boomi execution runtime as known as Atoms. Atoms are currently hosted by Dell thus Disaster Recovery is managed by Dell OPS team. Projects are notified by DIF team and Dell recovers the failing servers.

Logging

Execution Records (Process Reporting equivalent) and Execution Summary Records (Realtime Dashboard equivalent) are available on the Michelin Splunk to monitor every process execution.

Custom logging can also be done using the Notification shape (Dell Boomi process function) and will be utilized to log any custom application specific logs.

# Non-Functional Requirements

Non-functional requirements are use case specific. They will be captured and documented in the Detailed Design document that is prepared for each API.

# Assumptions

1. Marketing Cloud to provide a JSON schema for B2B and B2C customer entity.
2. The Apostrophe, Marketing Cloud and the API team will together define the intial mapping specifications to come up the Experience APIs.
3. The API squad will develop amd maintain the Dell Boomi APIs that are developed as part of this integration.
4. The Dell Boomi platform support will be managed by the DCSI DIF support team along with the Dell OPS team for the public atom.

# Dependencies

1. Required Dell Boomi atom will be available for the use of this project.
2. The performance and scaling of the APIs will be dependent on the Dell Boomi Atom.
3. The support and monitoring of the Dell Boomi Platform is performed by the DCSI DOF Team. Since it’s a SAAS solution, Dell provides the underlying monitoring and support as per Michelin contract with Dell.
4. The support and monitoring for the APIM is performed by the DSCI API Support Team

# Open Questions

# Appendix Section

**Conceptual Architecture view:**

The conceptual API-led view defines the main building blocks of the architecture, along with their purpose and responsibility. As in Figure, the conceptual API-led architecture consists of four main building blocks, plus the Consuming Applications. Horizontal blocks represent core runtime capabilities, without which implementing APIs would not be possible. Vertical blocks represent important supporting capabilities geared towards life cycle support, management, operations, and analytics.



Consuming Applications are considered as any computer program capable of calling and making use of an API. In real-world terms, these may vary from traditional commercial off-the-shelf applications (for example, commerce systems), to web applications, mobile applications, wearable devices, and even more sophisticated things such as drones and smart cars.

The API Exposure building block, as the name implies, is responsible for securely and reliably exposing access to API endpoints.

Services are units of software that deliver well-defined and bounded functionality. Such functionality is referred to as a Business Capability because it means something to the business and the function the service delivers can be mapped to a business process. Therefore, in order to deliver such functionality, a service must be capable of implementing business logic, data transformation and validation, business rules, orchestrations and/or choreographies to name a few. Services expose their functionality via API endpoints that are not accessed directly but mediated via the API Exposure layer.

The Management & Operations block consists of capabilities in aid of the end-to-end life cycle management of APIs, including, but not limited to, API design and mocking, policy implementation, deployment, promotion, runtime operations, and analytics, deprecation and retirement. Developer-centric capabilities such as API pages, a developer portal for API discoverability and subscription, and application keys management also form part of this building block. These building blocks act as aids when monetizing APIs, as it's responsible for collecting important metrics that might be required when billing APIs based on usage.

Lastly, the Identity and Access block refers to capabilities in support of users, roles, and access management features. For example, from a life cycle and operations perspective, this block enables different users (API product owners, administrators, designers, and developers) to seamlessly log-on to the API management console and/or API developer portal using existing enterprise credentials. It also restricts access to different areas depending on the user role. From an API Exposure perspective, this block aids authentication and authorization policies by, for example, enabling tokens (for example, OAuth 2.0, OpenID and/or even SAML) to be generated and enforced at runtime.

**API-led connectivity**

Doors are used as a metaphor to articulate the role of APIs in delivering access to enterprise information assets and functionality, or in business terms, business capabilities. However, just like doors, which come in different types, materials, and sizes often determined by what sort of access they provide, APIs can be classified in different types as well.

For example, some APIs might be built with a specific use case in mind and in support of a specific application. Because of this, such APIs can be quite specialized and tailored for the purpose they were built to serve. In order words, they are of single purpose and not suitable for reuse outside the context they were built for.

A common term used to refer to these types of [single purpose] APIs is Experience APIs, mainly because of their role in enabling applications that humans directly interact with (for example, mobile apps, web apps, and so on). However, not all applications that require specialized-purpose APIs have to interact with humans. For example, in Industry 2.0, APIs may be built in support of modern industrial lines, or in farming and agriculture, drones are being used as a means to scan soil conditions across large areas of land, and APIs are used to obtain and send data in real time.

Other APIs, however, might be built specifically with reuse in mind. Such APIs will be more generic in nature and won't be tied to a particular use case. Therefore, these APIs are multi-purpose, meaning they can be used in a variety of scenarios and thus should be able to serve many applications. The following diagram illustrates that APIs that don't provide access to tailored business capabilities can be consumed by many applications to address different use cases.

