zennify

A Modern Use Case for Salesforce Integration with API-Led Connectivity

by Sam Kerr

zennlab

Abstract **3**

Problem Statement **3**

Background **4**

Solution **6**

Setup and Configuration 7

Use Cases Solved 10

Comparable Salesforce Features 12

Conclusion **12**

References **12**

Abstract

Today [late 2019] Salesforce and its extensions like Financial Services Cloud (FSC) introduce a larger need than ever for unifying data from various external systems into a single 360 view of a customer. The concept of a CRM has transitioned into more of a customer experience platform than just relationship management and providing a great customer experience usually relies on data from many systems. The requirement for having real time access to externally sourced data within one system for a unified experience is in high demand. For Banks and Credit Unions, there is always the need to feed their member data to/from a centralized bank core. For Mortgage, there is always a need to integrate with a loan origination (LOS) or payment system. For Insurance, it's all about feeding data in/out of Carrier systems. All these verticals also share the need for other integrations on things like credit checks, background & financial checks, or really any system to help source data for the underwriting of a customer. FSC provides the framework for the data model and user experience tailored to these business verticals, but the actual connections to any external systems that feed the source data for this are left to the customer to choose, implement, secure, and maintain. Building this experience in Salesforce usually requires custom code to be written for each integration and introduces a level of technical debt in that the code needs to be updated whenever an integration or system changes.

Problem Statement

With the recent acquisition of MuleSoft, their new buzz is this "API-Led Connectivity" approach which can help make Salesforce integration much easier to manage and more scalable. Building point-to-point integrations from Salesforce to all systems is not a scalable approach anymore and the future is moving in the direction of API-Led connectivity where data can be requested and accessed on demand as needed and to build layers of APIs that de-couple processes to make reusable microservices. MuleSoft plays a big role in exposing data from these external systems and

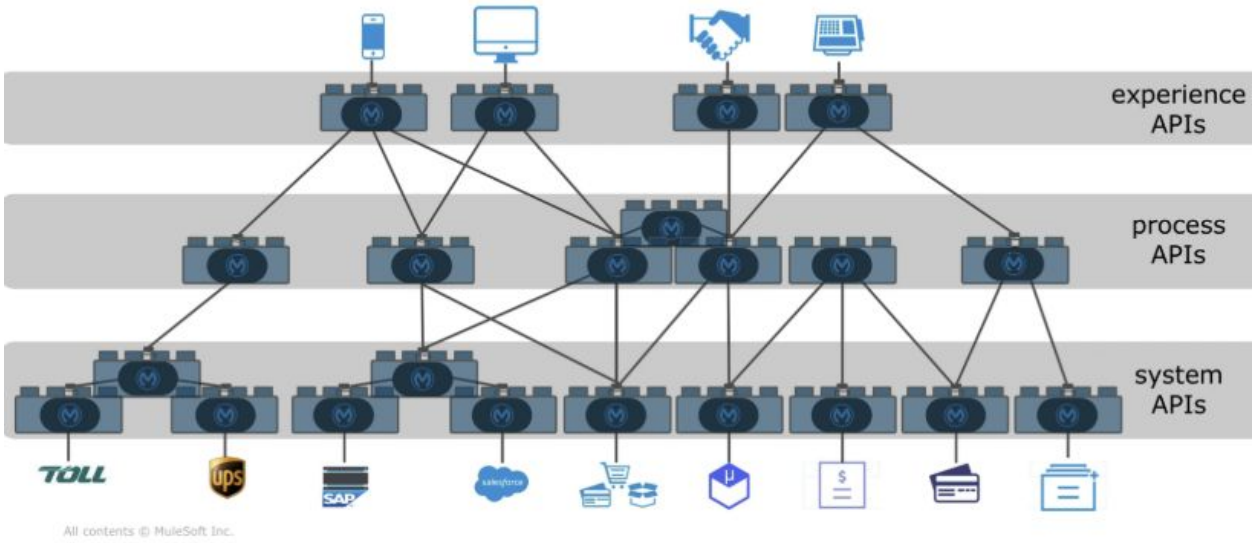
enables Salesforce to communicate. However, there needs to be a declarative approach to setting up the integration within Salesforce and mapping data elements to an API provided by MuleSoft (or any other system). This would eliminate the time spent in Salesforce making code changes, test coverage, and deployment each time an API changes or new fields are introduced which happens often.

There are many other middleware, ESB, or custom solutions, but this writeup will reference MuleSoft as the primary example. External data sources are not always cloud based with nice REST APIs for integrators to consume directly. MuleSoft has an extensive set of connectors to talk to APIs, databases, file servers, event streams, VPN tunnels, etc. This is a broad suite of tools to make data exposed and consumable. However, MuleSoft can't do all the work of detecting and feeding data directly into Salesforce whenever it changes in real-time to any connected external source without introducing performance issues or a major impact to Salesforce's daily API usage limits. Remember, Salesforce counts the number of inbound API calls towards a 24hr daily limit based on the number of licenses in your org (this limit can be seen on the Company Information page within Setup). However, outbound HTTP/s calls from Salesforce to an external service do not count towards a daily limit and can be done as many times as you'd like. The solution approach in this document takes advantage of outbound calls to leverage an API-Led approach for obtaining on-demand access to data in "real-time".

Background

API-Led connectivity promotes reusability of system integrations and sets up an organization for scalability as new systems are introduced or swapped out. No more point-to-point integrations that are proprietary to each system and developed in different tech stacks that require tribal knowledge of a legacy system. System APIs are developed at the lowest level to create REST API for interfacing directly with a specific type of system no matter what type of technology it implements. Process APIs

can then sit on top of that layer to orchestrate and aggregate data across many systems. At the top level are “experience” APIs that are built to serve a unified and easy to consume data model for a specific type of end consumer. Each of these APIs are reusable assets for any future integration to leverage.



MuleSoft API Layers

Layer	Ownership	Frequency of changes
System layer	Central IT	6-12 months
Process layer	Central IT and Line of Business IT	3-6 months
Experience layer	Line of Business IT and application developers	4-8 weeks; more frequently for more mature companies

MuleSoft building blocks for API-Led Connectivity (from “API-led connectivity for government”)

Salesforce can play a part in both a System API as well as an Experience API to act as either a producer or a consumer of data. The main use case that will be covered in this document is where

Salesforce is acting as a consumer to an experience level API. With this structure, an experience API can be developed to serve data to Salesforce for on-demand access of data it needs from any system and no impact to daily API usage limits. A data integration team could surely produce this API stack using MuleSoft, but then how can Salesforce best consume this and be adaptable to changes? There should be a repeatable and declarative approach to integrating Salesforce with a REST API without writing custom Apex code to query and marshal the data into required API structure and make HTTP callouts every time.

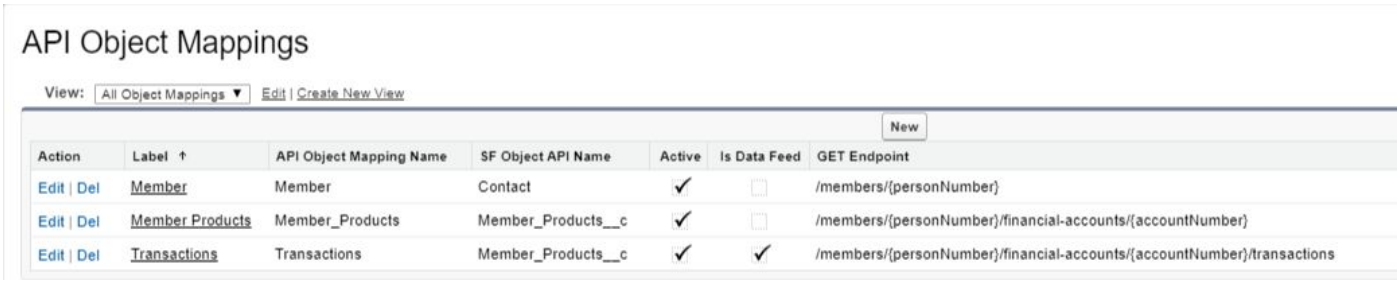
Solution

The API Integration Utility developed by Zennify is a native Salesforce managed package that provides a declarative mechanism using standard features for integrating with any REST API and provides lightning components for solving two main uses cases. The first use case is for on-demand data synchronization upon page load of a record. The second use case is for displaying rows of data from an external system with the appearance like they are Salesforce records. This utility works very well in conjunction with Mulesoft to consume an experience API that provides aggregated data from underlying systems.

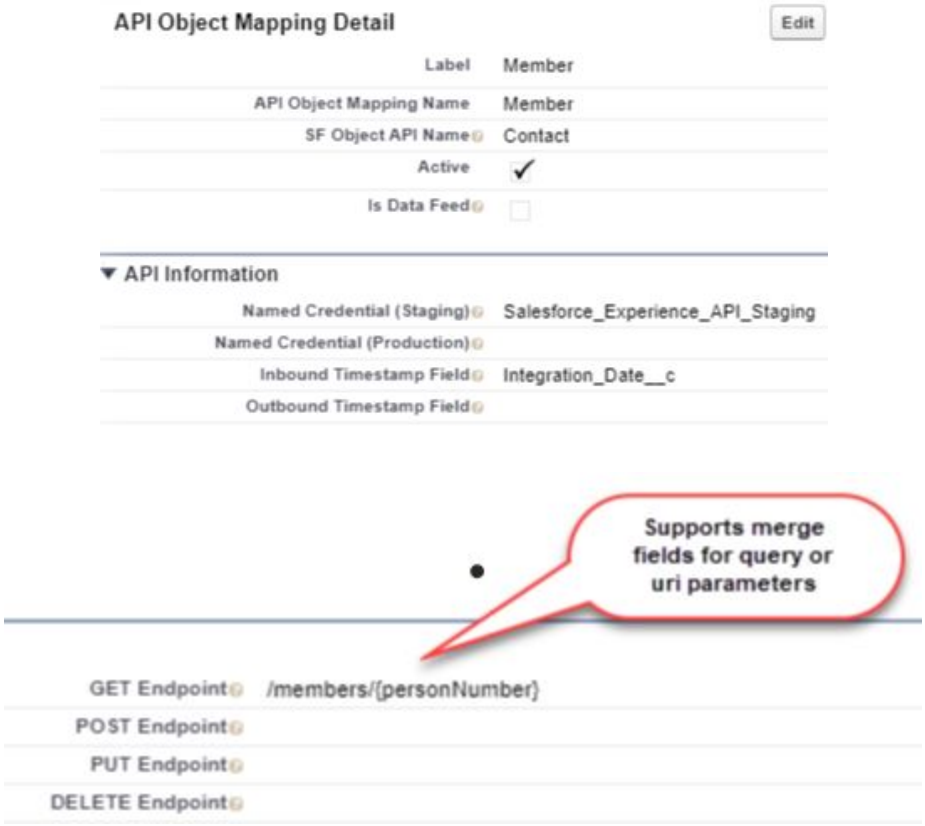
Setup and Configuration

The integration utility is driven off of two Custom Metadata Types and is where most of the configuration is performed. API Object Mapping is the first type and record of that will define for a given Salesforce object, what API might it integrate with and what resources will it need to access on it. It also defines if it will be used for syncing/replicating data into Salesforce, or if it will be used to feed and only display data in Salesforce that is not stored there. The other type is called API Field Mapping and is a child of the API Object Mapping. It defines every Salesforce field that is involved in either the HTTP request or HTTP response and how it maps to the model of the API resource. There are many configuration options at this level for how that field is used for syncing or being displayed. Creating records in these metadata types defines how an integration will operate. The next step is to setup the Lightning Page in use for the corresponding object and drag in one of the included components based on which use case is needed. Once completed, you have an API integration up and running.

Example list of object mappings:



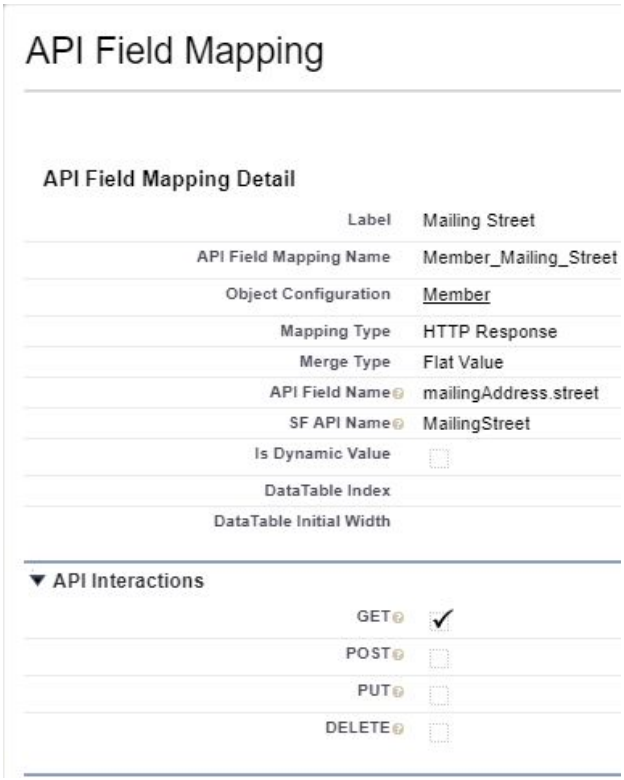
Example object mapping (specifies credential and endpoint details):



Example API JSON Payload with contact information:

```
{
  "customerId": "P123456",
  "memberNumber": "",
  "personNumber": "123456",
  "firstName": "James",
  "middleName": null,
  "lastName": "Bond",
  "homePhone": "(007) 123-0007",
  "mobilePhone": null,
  "workPhone": "(007) 456-0007",
  "email": "bond@james.com",
  "mailingAddress": {
    "street": "333 N Rodeo Dr.",
    "city": "Beverly Hills",
    "state": "CA",
    "postalCode": "90210",
    "country": "USA"
  }
}
```

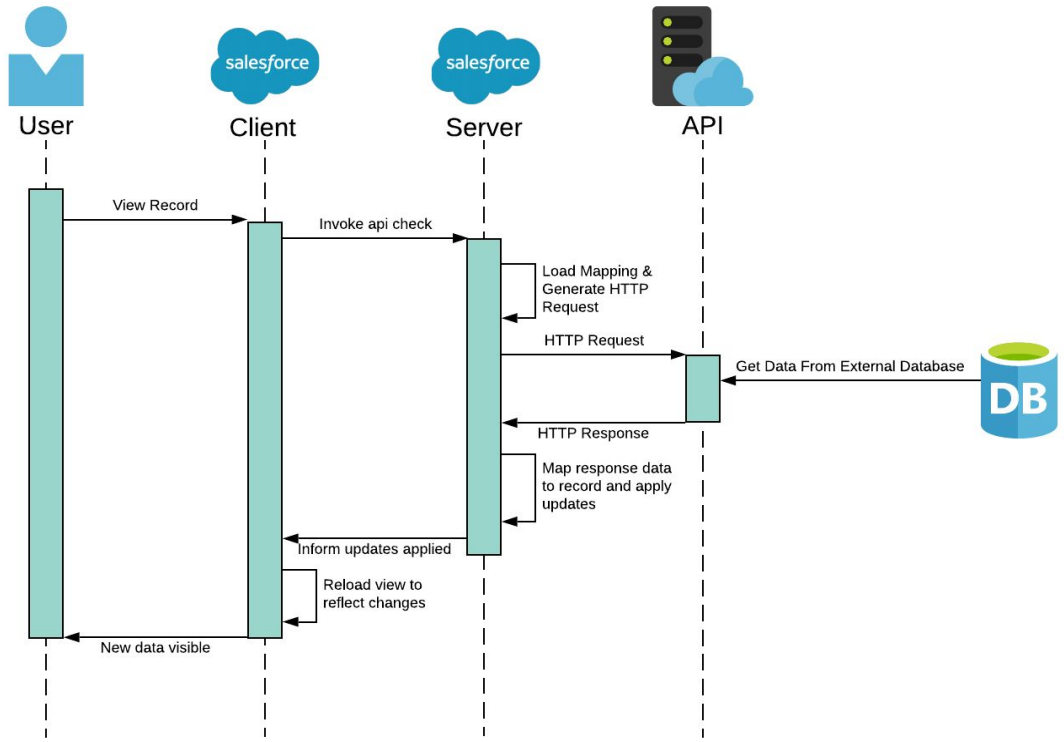
Example Field Mapping for obtaining Mailing Street Address:



Use Cases Solved

API Sync

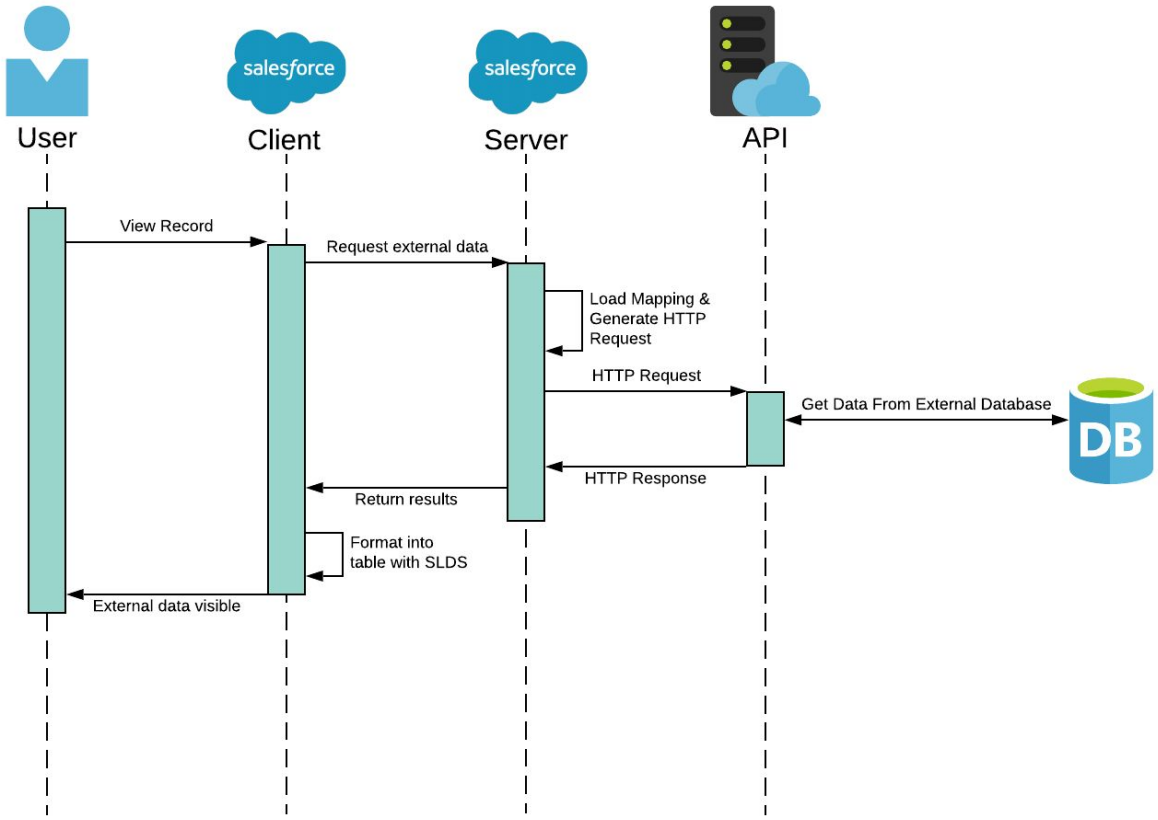
The API Sync lightning component is used for replicating data received from an HTTP Response of the configured API resource endpoint into Salesforce. When this component is on a Lightning page and active, the sync process is invoked whenever a record is viewed by a user. Behind the scenes, custom Apex within the package makes an HTTP request to the API and then compares each mapped field from the configuration and what is received in the HTTP Response to the corresponding fields in Salesforce queried from the record the user is viewing. Any data changes from the API are accepted and written to the record in Salesforce. Once complete, the API Sync lightning component is notified and refreshes the view of data on the record the user is viewing. To the user, they see the data update before their eyes without refreshing the page.



API Sync Sequence

External Data Table

The External Data Table component is for the other use case of displaying row of data in Salesforce formatted in a sortable lightning table. It includes options for pagination and search filters if the API supports it. This component was originally designed for, and works best with, displaying monetary transactions from a banking or financial system and is a typical use case of the type of data that is not usually stored in Salesforce but important to view within.



External Data View Sequence

Comparable Salesforce Features

- External Services - Auto-generates invocable apex for use in Flow for an API. Only works with APIs that implement Swagger Open API or Interagent schemas. No ability to reload data on screen without manual page refresh.
- External Objects - Interact with external record in Salesforce like a normal record. Includes many features like list views, reports, record view detail, pagination. However, this only works if you have an OData provider implemented on top of your data source (Heroku Connect makes this very easy). This also requires the OData 2.0/4.0 adapter license from Salesforce.

Conclusion

To summarize, Salesforce and MuleSoft are advertising API-Led connectivity and many organizations are going through a digital transformation developing this type of structure for their future. This framework promotes faster turnaround on exposing data from internal systems and future changes made to APIs, such as new fields introduced. Salesforce currently has no out of the box mechanism to declaratively build an integration to API driven data sources. The API Integration Utility developed by Zennify aims to fill the void of this gap to make declarative integration possible and act as the glue between MuleSoft (or any API) and Salesforce CRM. It also aims to complement experience level APIs produced by Mulesoft to give end users real-time view of core system data within Lightning Experience. The end result is a faster turnaround time in initial implementation and promotes easy maintenance for Administrators through declarative field mapping.

References

- https://www.mulesoft.com/ty/wp/government-api-led-connectivity