API Layered Architecture

API-Led Business Model

Today, in the age of APIs, an API is not just a technical interface on top of a database. On the contrary, your API is your new business model.

In the past, APIs were just seen as tools for developers. But nowadays, their scope is not limited to internal use; API makers are exposing their APIs for external users around the globe.

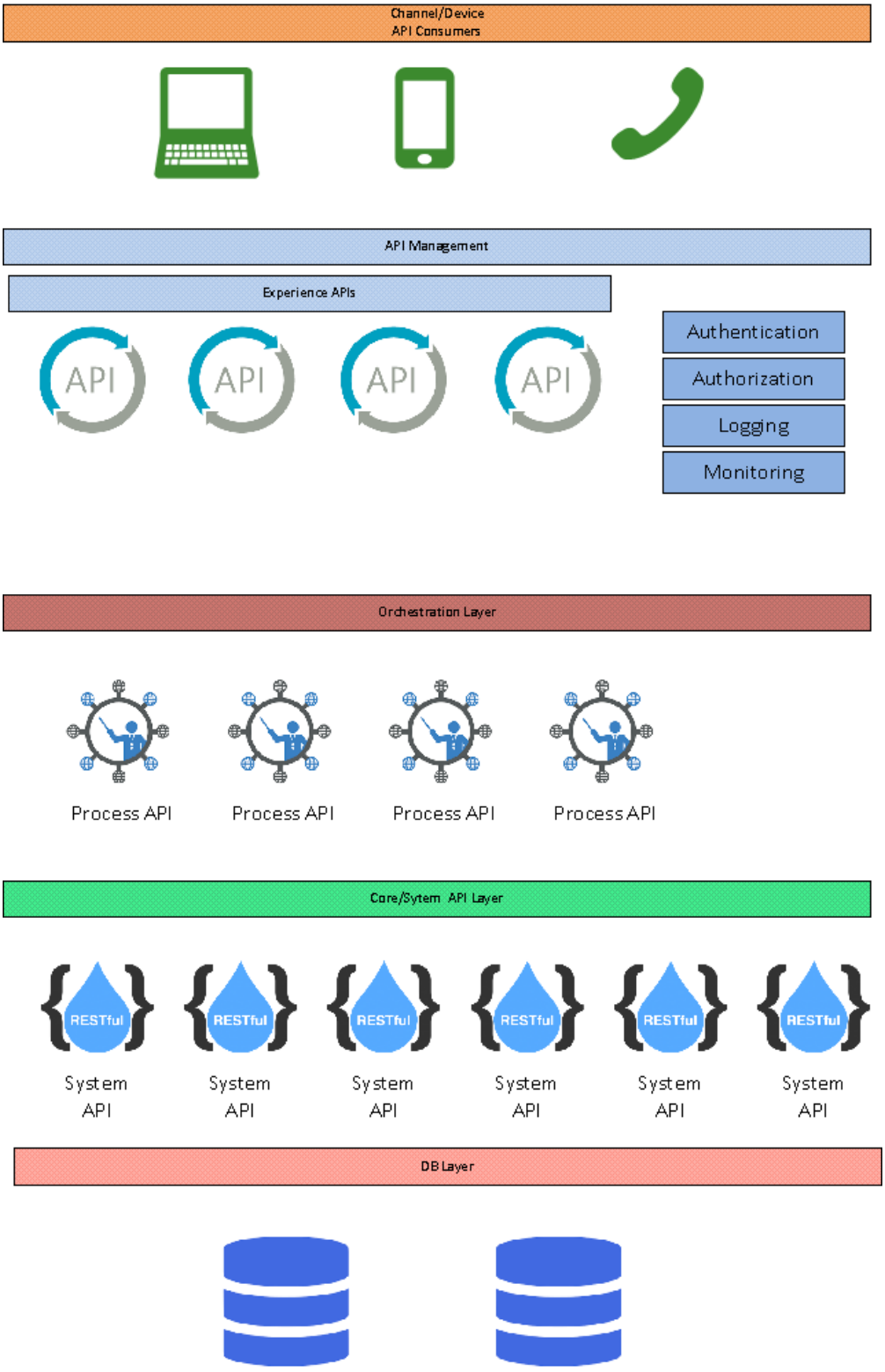
API Layers

The most efficient way to achieve API-led connectivity is to enable the integration flows to be reused by many parties and to be reused inside the integration platform**.**

With the re usability of the already available logic (implemented in flows), the developers can evolve their logic in faster and safer ways, leading to a short time to market.

APIs are created in layers and the best plus point as compared to E2E approach is that more components (flows) can be reused which makes easier to implement new systems and services.

In this approach, the APIs are based on three distinct layers: System, Process and Experience. From an infrastructure point of view it should look more or less as shown in the diagram below.



System Layer

This is the base layer of the three-layer API architecture. These are domain level APIs.

System APIs provide a means of accessing these underlying systems of records and exposing the data in canonical formats.

For example, a System API for customer domain will have CRUD (Create, Read, Update, Delete) operations exposing all customer specific information resources.

These are also called *Core* or *Enterprise* APIs.

This can contain sensitive information, and should never be exposed for public consumption.

Process Layer

Process layer APIs are responsible for shaping the data by orchestrating and choreographing various data by calling multiple System APIs. The orchestration involves the aggregating, splitting, and routing of data. The main purpose of Process APIs is to strictly encapsulate the business process independent of the source systems (System APIs) from which the data originates.

For example, to open a new acount, it needs to interact with customer API as well as Account API to successfully create the acocunt against the account holder.

The Process APIs should be held privately inside the organization, and should not be exposed for public use.

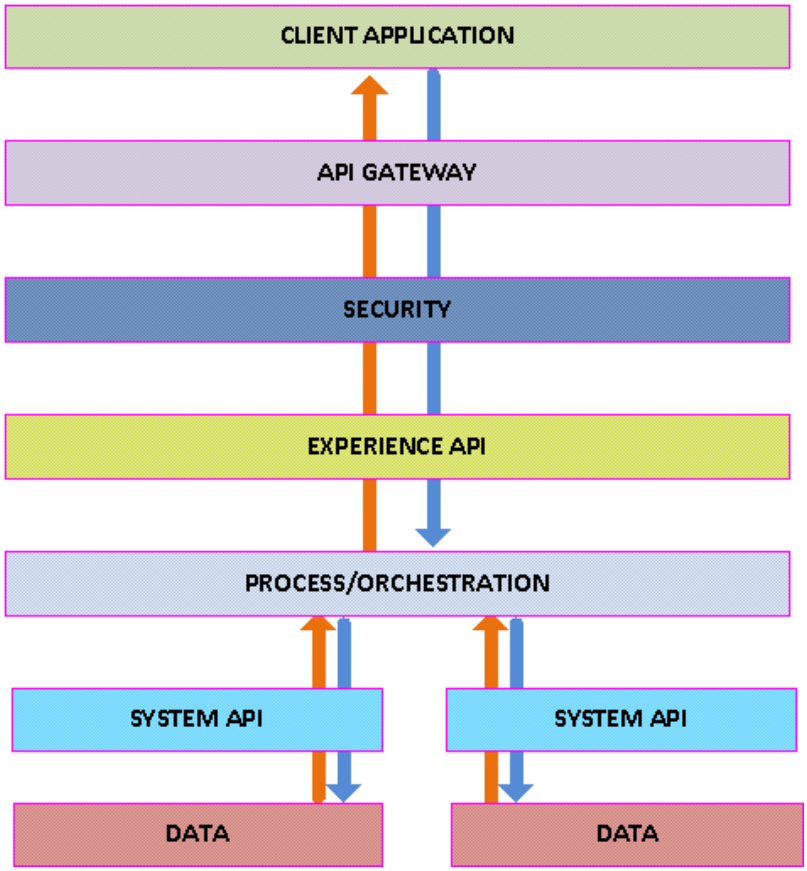
Experience Layer

At this point, we have all the sensitive information of an organization exposed privately by System APIs, and the Process APIs have already exposed the orchestrated business process. The business process data is consumed across a broad range of clients/channels with different formats. For example, in our previous example, the new account can be created over in a branch, which needs a different level of security, than if it's exposed to a TPP who has used sightly different authetication mechanism. With an Experiece Layer, it can offer two different set of APIs with varrying degree of security access. Also, there would be instances, where an expereince API would be very much channel (Mobile App, Tablet, WebSite, Smart Tv) specific and expose different models depending on the target consuming channel.

In other words, Experience APIs are the means by which data can be reconfigured easily to meet the needs of multiple audiences. Also, we can remove the unnecessary methods and expose only the necessary methods in Experience APIs in a simple way.

The Experience APIs are the ones which should be exposed publicly for consumption. In short, they are the final products of an organization in the API-led connectivity approach. Various policies can be applied to the APIs as well, as they can be monetized to earn revenue for the organization.

Below is a flow on how a typical API call would move through the flow


CLIENT APPLICATION
API GATEWAY
SECURITY
EXPERIENCE API
PROCESS/ORCHESTRATION
SYSTEM API
SYSTEM API
DATA
DATA