White Paper

API Driven Development

With the dawn of cloud computing and proliferation of apps, companies are exchanging data and services at an ever growing rate. APIs can increase agility by de-coupling and exposing business processes. The past few years, however, have seen such explosive growth that the API space is evolving more rapidly than ever before. In 2015, as many as 40 APIs were being added per week to the Programmable Web directory, and the total number of APIs stood at around 15,000.5 The key thing to consider here is that these numbers are based on publicly available APIs and do not reflect any private or internal API growth at all, which some estimate may even outnumber the public total. The future RESTful APIs will drive not only the exchange of data but also influence enterprise architecture.

APIs Transforming Business

Application program interface (API) is a set of routines, protocols, and tools for building software applications. An API specifies how software components should interact. Additionally, APIs are used when programming graphical user interface (GUI) components. A good API makes it easier to develop a program by providing all the building blocks.

APIs are transforming the business landscape today by - bring in standardization of interfaces in the development process. Developers get to work on structured and standardized APIs that are bound not to change their underlying behavior, irrespective of the technology or components used underneath.
APIs also take care of hiding the complexity of underlying implementation, bring in modularity and separation of concerns, which lets independent decoupled services to be implemented and tested.

Defining API Economy

API economy (application programming interface economy) is a general term that describes the way application programming interfaces (APIs) can positively affect an organization's profitability. An API is a customer interface for technology products that allows software components to communicate.
There was a time when only software professionals knew about APIs. Today, business leaders are aware of the financial impact APIs can have and companies are generating revenue by exposing APIs as business building blocks for third party applications. While revenue generation

is an important part of the story, the impact of APIs goes much further into organizations, enabling transformation and agility at many levels. APIs enable enterprises to deploy apps quickly, in a repeatable way,which leads to a faster pace of delivery, and the ability to create new and innovative experiences quickly. In addition, APIs can greatly reduce the cost of change, enabling IT and application owners to change apps with minimal impact – especially when there are numerous back-end integrations involved. This is critical to agility since the pace of change of the front end applications is much faster than in the back-end applications. APIs also help enterprises achieve operational efficiency,enabling greater visibility and expanded capabilities since every API call from the mobile app to the backend system is tracked and traced through an API key.

API Driven Development (ADD)

ADD in its current form is expected to have the following characteristics

- An API First Design where the API is the first artifact that is to be created during the app development process. API contracts (API Specification/Signature including the name, parameters, types etc) are either created by dedicated API architects or by front-end developers, who are responsible for creating the end user experience. API contracts are finalized in collaboration with front end and back end developers

- Once the API contracts are finalized, the Front end developers build mocks around APIs and create and refine the end user experience. In parallel, the back end developers implement the underlying logic of the APIs. Dedicated test suites are created around these APIs and in a way foster the idea of Test Driven Development. Finally the implementations of the front end and back end developers are brought together. This is bound not to fail as long as the developers have coded honoring the API contracts as established in the first step.

- At a code implementation level, APIs these days are typically designed using the REST architecture with JSON payloads. SOAP, XML and other standards are found to be heavy and going towards oblivion.

Benefits of API Driven Development

API Driven Development adds its own set of benefits to make the life of a developer easier and the process of app development simpler:

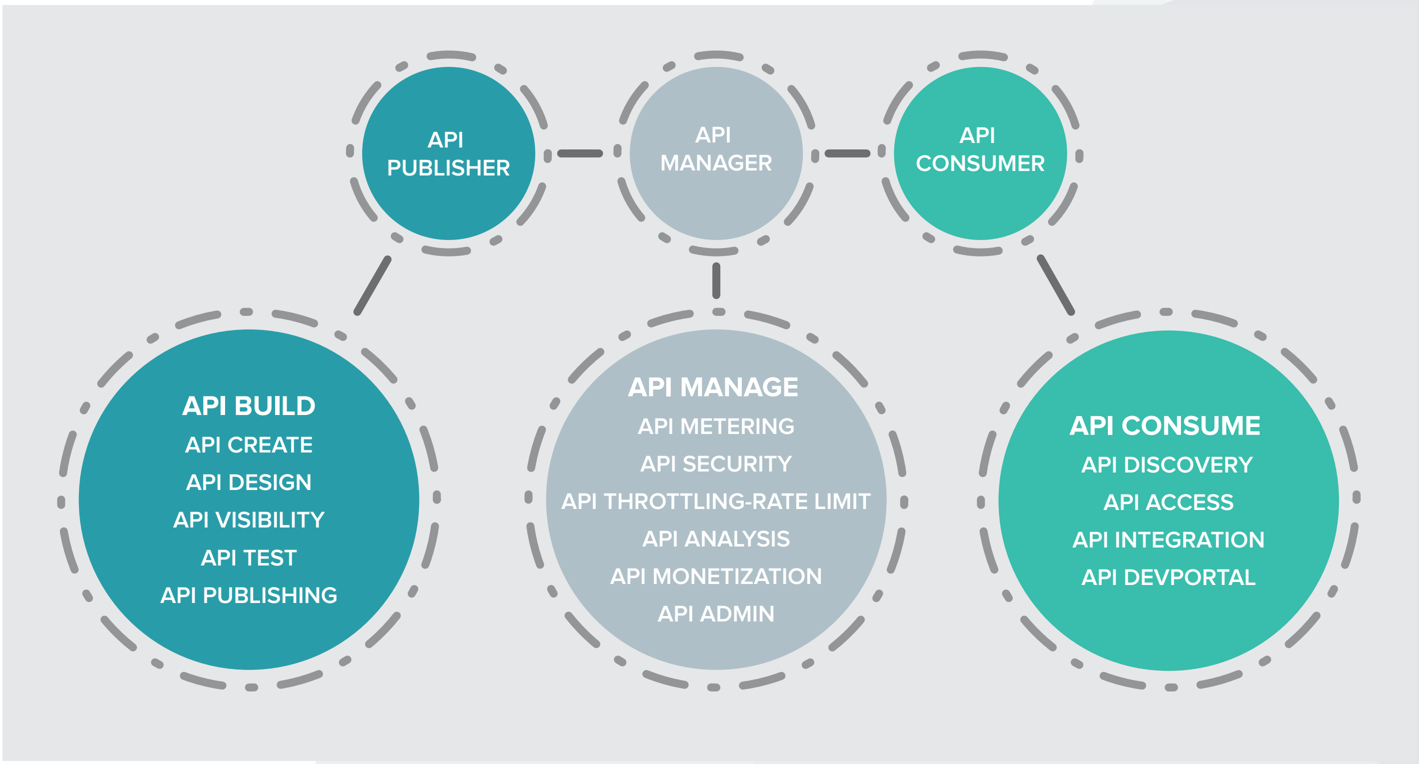
- Faster App Development
- ADD allows the developer to focus only on the business logic. In ADD, it is expected that all subsidiary services will be accessed through APIs. Also the initial agreement of the API contracts, parallel implementation by the front and back end teams and hassle free merge of the front and back end code brings about a huge time savings in app development time Focus on just your business logic
- It is expected that the developers will use APIs (3rd party/internal-cross-team) for all the subsidiary services like user management, logging etc, so that the focus of development is purely on the implementation of business logic and not on spending time implementing the skeletal structure of the app. In fact, in the thriving API ecosystem, API marketplaces (like www.mashape.com) have come up to make the process of discovering and consuming the required APIs much more easier.
- Better documentation
- One thing you will observe in all the enterprises who have made it big in API economy, viz Twitter, Expedia etc, is that their APIs are easy to use. Their APIs are easy to find and have great documentation surrounding their APIs. An effective API is defined by an effective documentation around it. In fact there are API tools like Swagger which have sprung up to aid in the process of describing and visualizing web APIs. In summary, a positive side effect of ADD is great API documentation, which typically is a neglected aspect of the traditional product development
- Inherently Micro Service Architecture (MSA) based applications
- One of the aspects we discussed in the "Why APIs"" section was modularity. Apps developed using ADD tend to be modular in nature, with every module representing a service (3rd party or own). The main application itself seems to be a collection of micro apps talking to each other using APIs. This app architecture is called MicroServices and has its own set of benefits. For example, If there is a load on the user management service of an MSA app, we can scale that user management service by adding more hardware resources. Compare this with traditional monolith app architecture, where the entire app
- Your app is ready for the connected world

- With proliferation of smart devices and the evolution of the API economy, we are heading towards a digital world where everything is connected through APIs. Forrester calls this as "APIs become the digital glue" and lists it as a top technology trend to watch for 2014-16. With API Driven Development having API as the fulcrum, your app is will not just survive but also thrive in this connected world.

API Management - Bridging the Gap between Developers and Customers

API driven development allows development teams to develop applications with APIs as the key focus. The APIs used by the application are given highest priority and developed to meet customer requirements. The API forms the foundation for the other application layers like the user interface, integration points, etc. Within the Agile framework, the APIs are developed in increments and are regularly reviewed by the customers and suggestions are made. This approach works well to ensure all customer requirements are considered.
API management solutions help ensure that partners and developers are productive. An effective API management platform also ensures that the entire lifecycle of an API is managed. Lets see what do we mean by API Lifecycle and management. The customer lifecycle is thus the most most important imperative

In API life-cycle there are 3 primary personas:

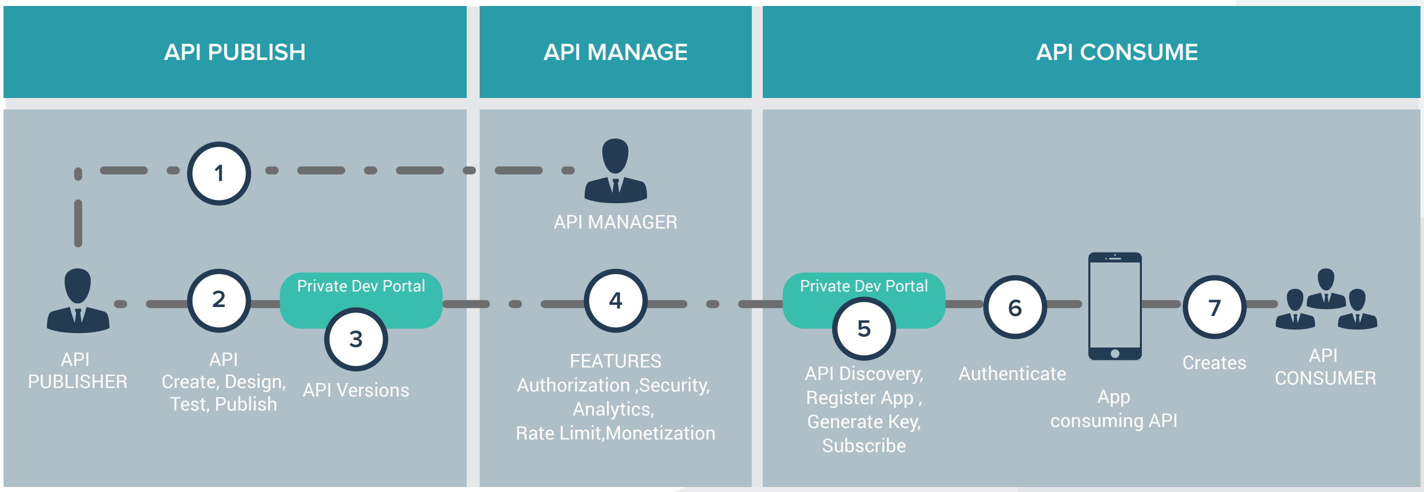


- API Publisher who creates and deploys the API
- API Manager who manages and monetizes the API
- API Consumer who discovers and integrates with the APIs

Each of these API personas have multiple tasks associated with them and those tasks define the characteristics of the API

What happens currently?

Let's take a look at the API life cycle in another dimension (see Fig-2), and understand the different stages in an API life-cycle. Here you would see how the various personas deal with APIs in different stages. Let's jump in...



Note: API Manager, API Publisher and API Consumer are personas who may be represented by people with various designations in an organization.

- API Manager, a persona who could be represented by people with designations such as API Product Manager or API Architect, prepares an overall plan on how to expose an enterprise's digital assets using APIs. The plan can include list of APIs, their design including parameters and their types, visibility scope etc. Last but not the least, API manager also makes sure that the documentation of the API is comprehensive. There is a popular saying in the API world - An API is good as its documentation. Hence API documentation is one of the tasks that should be done with utmost clarity.

- Once the API plan is ready, the API Publisher, who can be represented by people with designation such as software developer or software architect, gives birth to APIs by creating them as a part of the core platform development process. Note: A lot of times you would find the personas of API Manager and API Publisher being delivered by someone like an Enterprise Architect or similar designated persons

- API undergoes further tweaking to its design configuration (eg. path or query parameter) as per the requirements, tested and published to the enterprise dev portal. Whether an API is private or public is defined by the APIs visibility settings and the governance semantics around it. You would normally see vendors using API frameworks such as Swagger and tooling around it to do the above tasks easily.

- API Manager can apply more management configurations based on requirements. For instance, API Manager may configure to allow 1000 API invocations free and the consumer has to be pay for invocations beyond that. He can also configure analytic reports on various APIs and can create plans that can be subscribed to by the consumer. At this point, all APIs are assumed to be published and available for consumption, through an API DevPortal (Check out Paypal's API dev portal by clicking HERE)

The above 4 steps (as also seen in Fig-2), explored the API life-cycle from the perspective of publisher and manager.
Now, we will explore the API life-cycle from a consumer perspective. API Consumer persona is usually an app developer who consumes or integrates the APIs as a part of the app development process.

5, 6, 7: In the API DevPortal, the consumer has the ability to search for the desired APIs, subscribe to a particular API or an API plan associated with a group of APIs, as defined by the API manager. The API Consumer creates a DevPortal account and registers his app to consume the needed APIs. Usually an API key is generated per app and is used during the run time to authenticate the app accessing the API. Consuming the API is the last step in the life-cycle of the API. When you are able to successfully integrate the API into your app and get the desired results, it completes the cycle.

Note: A few API Management vendors use another life cycle step of engagement/promote where the APIs are promoted in various forums to the end consumers(who are developers). I have not touched that aspect of the life cycle, since it may not be that relevant from ADD perspective.

How Low-code Platforms are ideal for API-driven Development

A good low code platform should be able to enable both the API Publishing and consumption and have solid collaboration with a API Management platform as well. Let's delve into the details now :

Auto generation of APIs

For sometime now ,Low code platforms has been auto-generating the code based on the visual development. It's time to do the same for the APIs. Some of the most common APIs that can be auto generated can include the services from DB, external services, custom coded business logic, security services etc. For instance, it's imperative that low code platforms, at a minimum, should auto generate CRUD APIs for the associated DB entities. More advanced platforms can also APIfy the SQL queries and DB stored procedures allowing total control for the users

Other services like security and custom code business logic are also great candidates for APIfication. For instance, if you have custom coded in your CRM app a function to take return the list of all users from EMEA zone, then that function should ideally be APIfied, automatically.

Automatic conversion of SOAP to REST

APIs these days are invariably REST based. But there are still big remnants, of legacy SOAP based APIs and modern low code platforms would automatically create a REST API endpoint for the app. This auto conversion is especially imperative in an enterprise setup, where legacy baggage are seen far too often. The automatic availability of REST APIs is an important step in modernizing legacy apps.

Easy consumption of APIs in an app

Out of the box integrations and connectors are increasing making its presence in today's market.and a lot of app Development platforms focus on them . But often these platforms do not realise that there is always a custom API requirement for an app. Hence all kinds of integrations - both out of the box and custom should be treated equally.

Ease of design, testing and sharing in APIs

Another pitfall in many low code platforms is that they tend to focus a lot on API consumption through connectors and forget about API publishing aspects. In a connected app world, it is imperative that your own app should have easy ways to create, design and share APIs. Inbuilt tools that can design your APIs (for eg, configure path parameters vs query parameters) with ease, test them (through an integrated testing sandbox), and then publish (private, public access) them are important features in any modern low code platforms. There should also be easy integration to publish these APIs into the enterprise API management platform, so that it's instantly available to the API consumers within an enterprise.

Overcoming technical hurdles of APIs.

Modern REST APIs, though simplified, are still quite technical in nature. There is still technology involved in understanding path versus query parameters, headers, auth headers, API key etc. Low code platforms, which are positioned as app building tools for business users and citizen developers still find it difficult to work directly with APIs. A smart low code platform abstracts these complexities and provides you with a nice UI based connector to work on. This is where the out of the box connectors become really helpful. But even APIs of your own app should also be abstracted the same way. That is where a 2-pass development approach would be very helpful in terms of creating reusable UI components for the business user.

API Driven Development is still just a wishful thinking than a reality among low code platforms. This is one space WaveMaker scores far ahead of its competition. This is one of the features that is going to make you ready for app building in the modern digital world.

About WaveMaker

WaveMaker Platform is built on the most open, extensible and flexible fashion to complement enterprise application delivery keeping in mind the requirements of a Software Developer, Citizen developer / business user, IT architect and CIO.

You can Get Started with a Free trial of WaveMaker Low code Platform.