Mindtree
A Larsen & Toubro Group Company

The Journey of API maturity

April 2020

#ReimagineNewNormal

Table of Contents

1. Introduction

In the current economy, organizations are striving to enhance digital capabilities to do business and ensure a smooth experience to ensure customer loyalty. Today, customers are empowered to seek out the easiest and most effective experiences. To win in this digitally connected world, you need a strong customer base along with an ecosystem of partners to fuel business growth.

Applications and Data are the cornerstone for running business processes, delivering personalized customer experiences and collaborating with partners. As the business and technology ecosystem has become more complex, Application Programming Interfaces or APIs have interestingly became the glue between the organizations, the consumers and their partner ecosystem. While APIs have been around for quite some time, they have not only increased in numbers, but have also become scalable and can be monetized. In other words, API integration is no longer just for developers, but has become a part of the boardroom discussion.

It is important to consider how organization interacts with all segments of their customer base, derives value from the surrounding ecosystem, and thinks strategically about integration. With APIs taking the center stage across the organizations, it is time for IT leaders to assess and improve their API strategy by looking at:

- Maturity of the API platform
- Maturity of the API implementation

Just like in earlier times - the kingdoms who knew the art of war would always succeed; in today's world it is the Art of APIs which would keep organizations ahead.

2. Glossary

Abbreviation	Definition
AI	Artificial Intelligence
API	Application Program Interface
B2B	Business To Business
C4E	Center for Enablement
CoE	Center for Excellence
ESB	Enterprise Service Bus
FAQ	Frequently Asked Questions
FTP	File Transfer Protocol
HA	High Available
HATEOAS	Hypermedia As The Engine Of Application State
IQ/OQ	Installation Qualification And Operational Qualification
IT	Information Technology
LB	Load Balance
LOB	Line Of Business
ML	Machine Learning
MVP	Minimum Viable Product
POS	Point of Sales
RAML	RESTful API Modeling Language
REST	Representational State Transfer
RFI	Request for Information
RFP	Request for Proposal
SDLC	Software Development Life Cycle
SLA	Service Level Agreement
SME	Subject Matter Expert
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
WSDL	Web Services Description Language
XML	Extensible Markup Language

3. Why API Maturity model?

APIs have been around for some time now as a way for IT organizations to offer services, connect systems and things, and adopt an omni-channel strategy and the hybrid IT paradigm. Over time, the API space has seen a lot of development and capabilities added, so much that organizations are now looking at API platforms not just for internal usage, but for leveraging it as an ecosystem to grow business and adding new revenue streams. With rapid growth comes significant time-to-market pressure, which shifts the organization's focus from being able to look at the API as a Product to API as a Project. This emphasizes the necessity to bring focus on building platforms with the right capabilities for it to cater to the future needs, rather than fulfill just-in-time project needs.

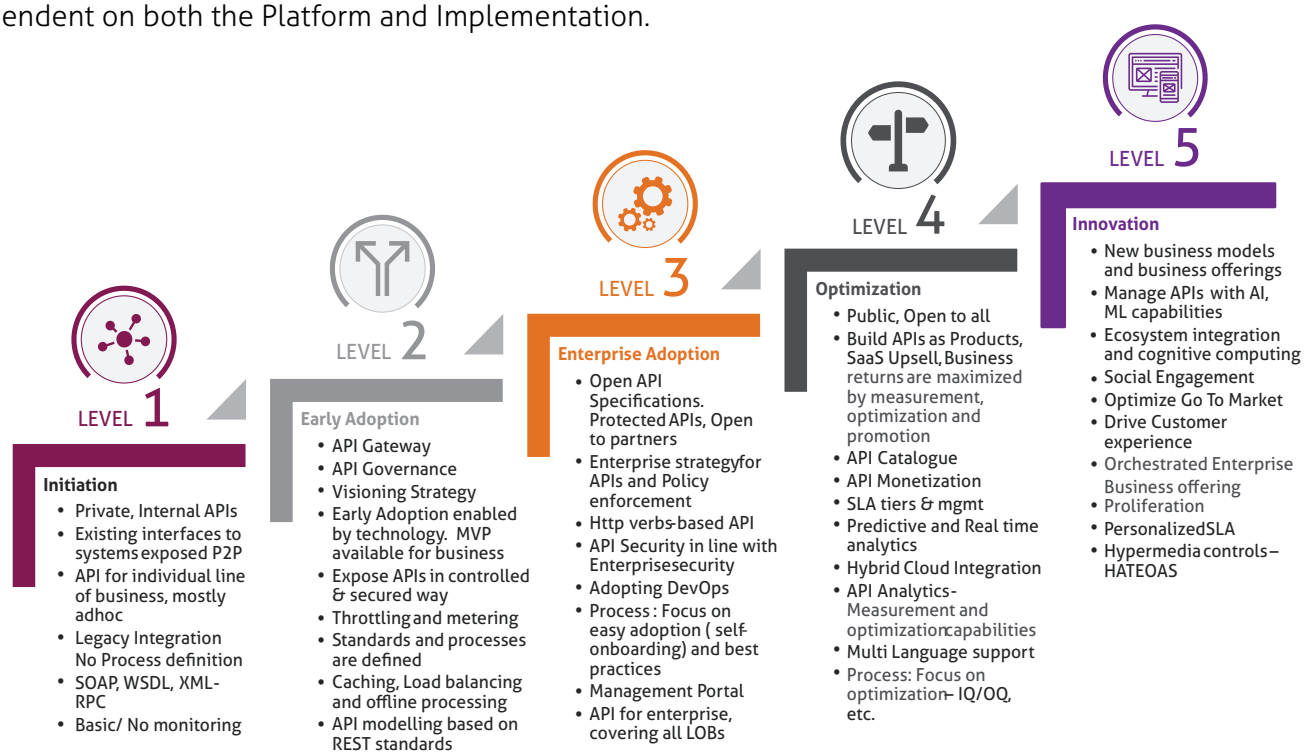
The transition from API as a project to product is a journey that most organizations are on. This requires an understanding of your current level of maturity and the steps to be taken to move to the next level. Here, API maturity can be assessed by looking at two key aspects:

- API Platform
- API Implementation

API Platform maturity is assessed based on the platform capabilities which ensures that an ecosystem is supported. It caters to below aspects.

Gateway | Governance | Throttling | Metering | Availability | Scale | Self-service | Analytics | Developer Portal | Automation Dev-Ops | Change management | and more. It's a way to ensure that doing business is made simple, effective and futuristic.

On the other hand, API implementation ensures the maintainability and readability of the API along with the adoption of API implementation standards. The maturity of the API setup for an organization is dependent on both the Platform and Implementation.



API Maturity Level

API maturity is a journey that starts from the initiation stage and is aligned to the transformation needs of an organization. It is a five step journey starting from the initiation phase, and organizations mature based on their API strategy, ongoing initiatives and progress made. The other way to look at this is that, the levels are an outcome of gaps between organization's needs and barriers to success. While the definition of maturity can vary for organizations, a few could be business-led or technology-driven, and/ or may have taken a mixed path. Based on our experience working in this space, we have observed that most organizations have initiated their journey with their API strategies and programs and are typically somewhere between Level 2 and Level 3 of the maturity model depending on the implementation state and timelines.

Needs

- Increase digital footprint and digital landscape
- Consistent & Seamless experience across physical and digital channels
- Rapid Co-creation and innovation with customers and partners (extended enterprise)
- Support growing ecosystem of devices and things
- Share digital assets across the extended enterprise.
- Expose Legacy Data and assets
- Self onboarding of Partners and third party developers
- Modernize B2B Platforms to API enabled platforms
- Access to Usage patterns and Key results.
- Nimble introduction & extension of APIs
- Customer Listening and understanding of behavior and interests.
- "Always on" solutions

Barriers

- Limited Digital Presence
- Constraints in reaching to customers across both physical and digital channels
- Barriers in accommodation of new channels
- How to share assets in a secured way with various stakeholders
- Bottleneck in Mining legacy data
- Challenge in measuring right result
- Imagination of the future road map
- Rigidness to change
- Low technical Capabilities
- Maturity across the organizations and business eco system
- Adoption of Automation
- Level of maturity and monitoring capabilities
- Identification of teams and roles

Let's deep dive to understand each maturity level and the differences between them:

3.1 Initiate

Organizations at this level are starting their journey with some initial use cases and are exploring API platforms. At this level, organizations are typically addressing the 'Why and How' questions about APIs. At this level, organizations would be dealing with a few of the below:

- LOB/ Business identification
- Initial use cases most of which are ad-hoc, building internal APIs
- Implementation of APIs which have mostly single or maximum of three consumers
- Implementation of few patterns and legacy integration use cases
- Basic or no processes in place, usage of only default monitoring options

3.2 Early Adoption

This level is about adding more features to the platform; strengthening the foundation with frameworks, governance, strategies (such as versioning, backward compatibility, and policies), adding more patterns and making the platform more capable. Organizations at this level would want to see what more they can do to support early adoption by businesses/ LOBs in ideating and building their MVPs. At this level, organizations would be characterized with few of the below:

- Ideating and realizing MVPs for businesses/ LOBs
- Standards and policies such as backward compatibility, versioning strategies
- Strengthening the platform with HA, LB, caching features etc.
- Implementing API-led modelling
- Building in processes, frameworks etc.
- Emphasis on documentation has started

3.3 Enterprise Adoption

At this level, the platform initiatives are more focused on the channels, end consumer and the usage of the API at an enterprise level. Organizations at this level are dealing with how they can improve the experience and provide secure access for the enterprise to increase adoption. It is necessary to strengthen the following to ensure ease of API usage and secure access:

- Initiate journey on developer experience, basic developer portal
- Customer facing APIs and open API specification
- Scale and security
- DevOps implementation
- Thinking of enterprise-wide API platform
- Collaboration, packaging and billing - all in place
- Processes to support self-service and other related best practices
- Introducing C4E (Center for Enablement) to the organizations along with associated guidelines for adoption

3.4 Optimization

Once the platform is setup with features, capabilities and automation, the organization needs to look at generating returns by optimizing abilities to measure usage, maintain implementation, promote APIs, and leverage the ecosystem. Organizations at this level are dealing with how to make the platform more optimized, measuring the usability of the platform, getting visibility in the implementation and helping users be more collaborative. At this level, you will most likely be dealing with few of the below

- Upselling APIs as products
- API analytics, add on to the basic analytics available
- Multi-language support through documentation and developer portal capabilities
- API ecosystem enhancement and leveraging the same
- Process improvements by adding measures for strengthening self-service and minimizing governance
- Moving teams from project to product implementation

3.5 Transform

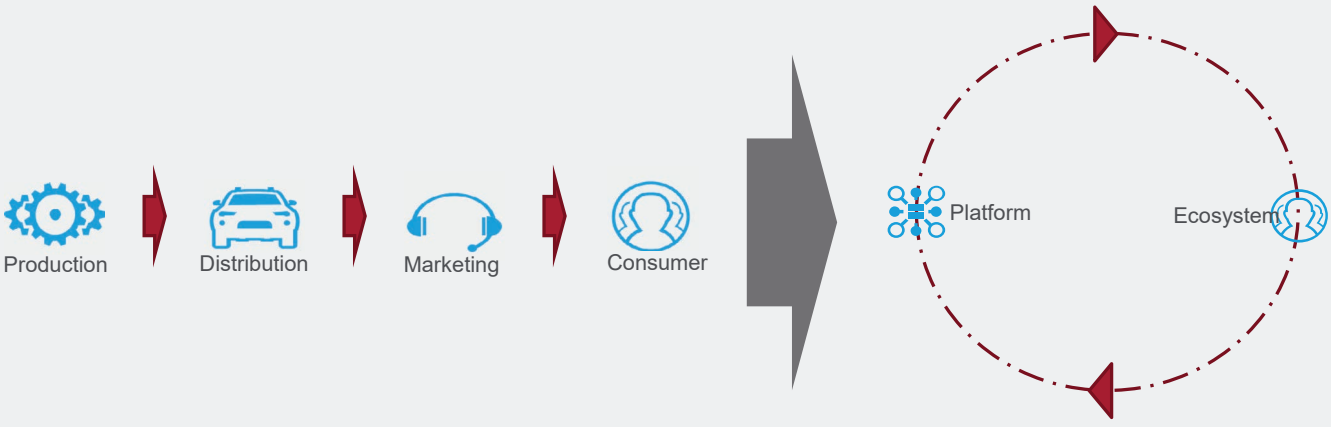
This is the highest level of maturity that drives innovation and ensures superior user experience with least possible friction. It drives greater transparency among users and organizations, offers personalized experience (if needed). Having said that, the governance, team structure and organizations moving from project to product thinking would be more mature. Organizations at this level are dealing with questions about what else can they do to increase the API maturity and the areas to innovate in. At this level, you would most likely be dealing with few of the below:

- Innovation either through technology "Hypermedia as the Engine of Application State" (HATEOAS), Process (Self-service and Bot automation), business initiatives (collaborative offerings)
- New business models and business offerings
- Manage APIs with AI, ML capabilities
- Ecosystem integration and cognitive computing
- Social engagement driving ecosystem
- Technology innovations driven with event-driven/ web hooks
- Personalization

4. Framework for measuring API maturity

The journey of API maturity would lead organizations to move from traditional value chain to being platform-driven. In other words,

Moving from **Linear and one – way value creation** → **Two – Way and Continuous**



The above transition is based on three derived laws of API.

- API usage - Which is related to experience
- Ecosystem - Which is related to community
- Revenue - Also referred to as monetization, an important reason why the API journey started in the first place.

3 Laws of API

API Usage α API Value/ API Friction

API Usage α API Value * API Ecosystem

Revenue α (API Value) $^{\text{API Ecosystem}}$

Examples for API value for domains like

- Airlines – API value could be customer preferences, booking, ancillary services etc.
- Banking – New products targeting newer age groups, Fintech companies partnering with banks etc.
- Pharma – Clinical trials, adverse events for research, API ecosystem for larger good

The above laws lead us to the next set of questions, which are

1. What is API friction, what does it affect and how can I address it?
2. What is an ecosystem, who are the users, how does it work?
3. What is monetization, is it internal or external?

While the above questions solve only one aspect of the API maturity puzzle, it is necessary that the questions below be addressed as well to ensure complete coverage.

4. What is the role of governance, how to structure API Team and what are the roles?
5. How do we encourage organizations to move from project to product mode?
6. How to drive maturity in monitoring?
7. How to address API security?

4.1 Friction

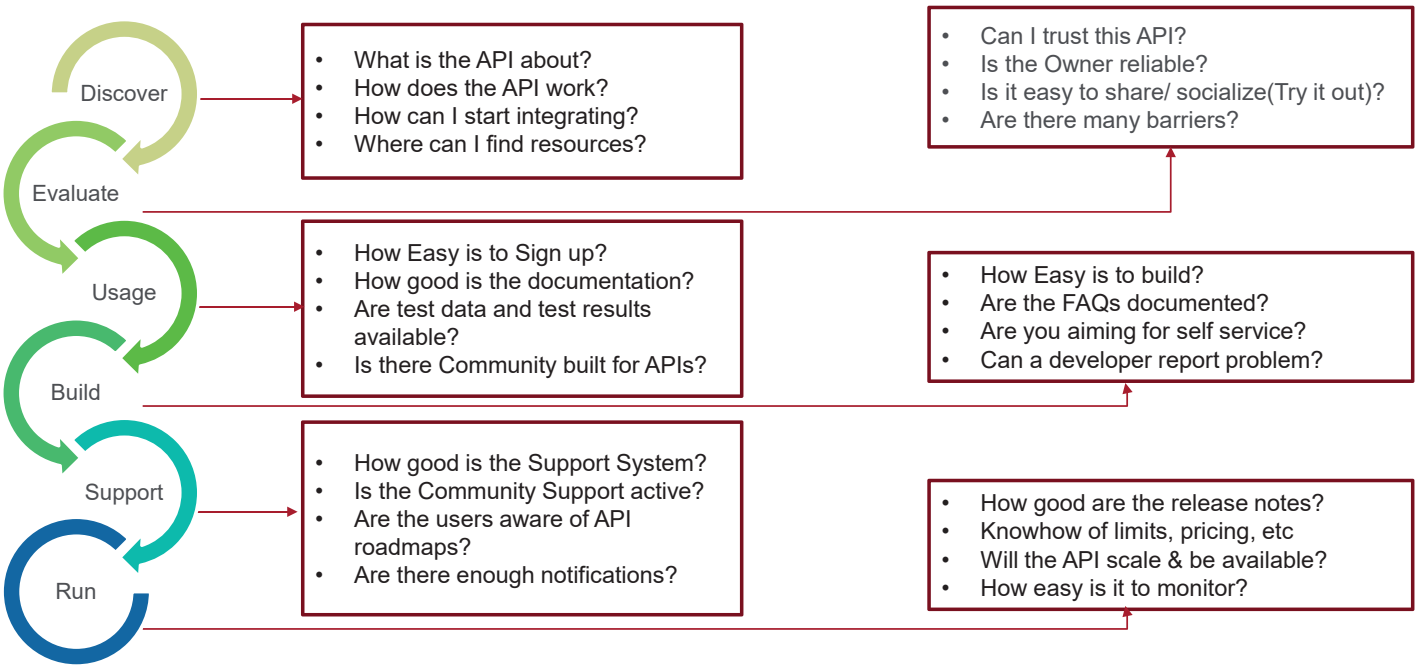
The concept of friction in products and applications is not something new and can be experienced by almost everyone. Friction in API is defined by resistance that an API offers to anyone who wants to use it. API friction is an indication of the level of difficulty in understanding and using the API according to developers.

Hence the 1st law, API Usage α API Value/ API Friction

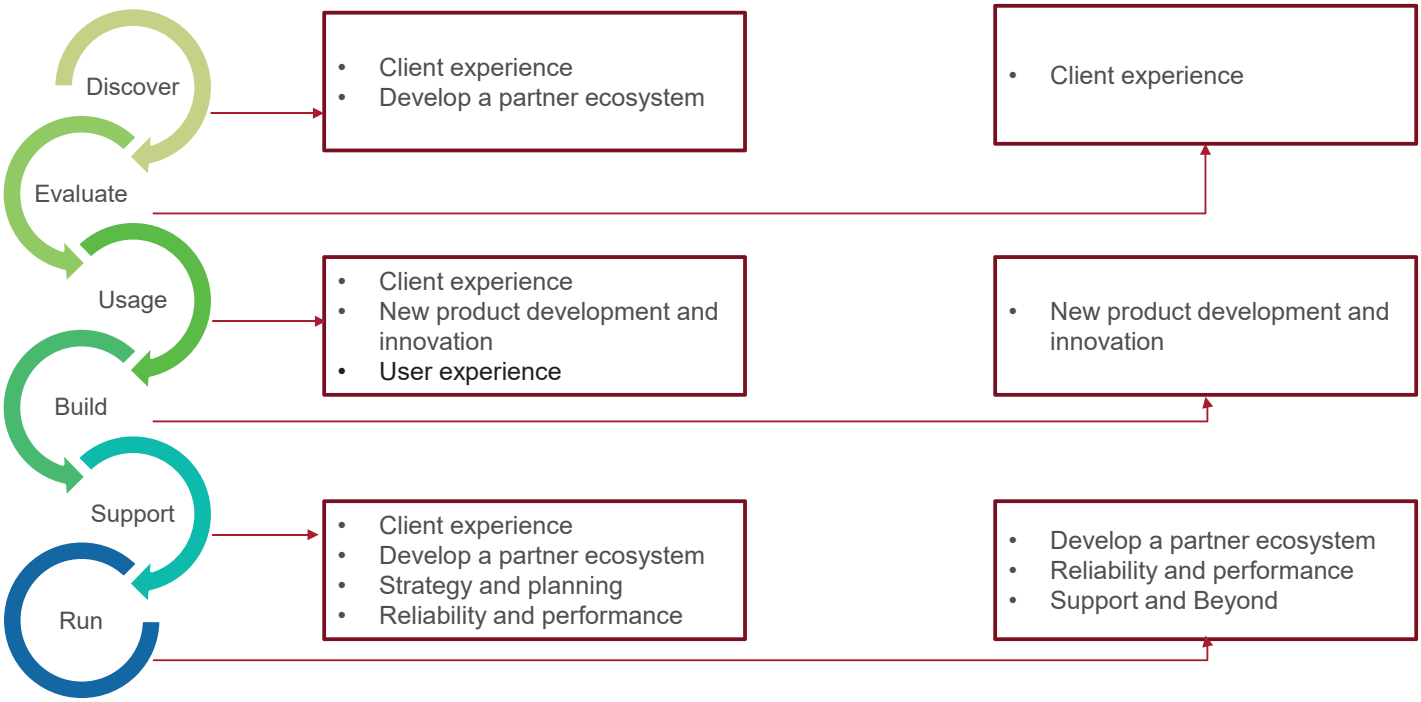
The API experience for developers increases as the usage friction reduces. There are multiple factors which could increase friction which includes:

- Understanding the API
- Testing the API
- Lack of documentation
- Technology issues
- Community problems
- Support experience

While the list can be long and can be associated with the API across the software development life cycle (SDLC), there is a lot which can be done to ensure friction is reduced, provided we ask the right question across each phase and act. Below is the SDLC for API usage/ consumption and typical causes of friction.



The friction introduced across the SDLC can influence the experience in some or the other way and reducing it can help in a long way.



4.2 Ecosystem

Once valuable APIs are built on a platform with the objective of increasing the reach of the organization's offerings and services, an ecosystem is necessary to drive further adoption. In other words, if APIs expose an organization's services/ offerings digitally, the ecosystem drives the usage of the APIs (Assuming friction is managed well or the API experience is good)

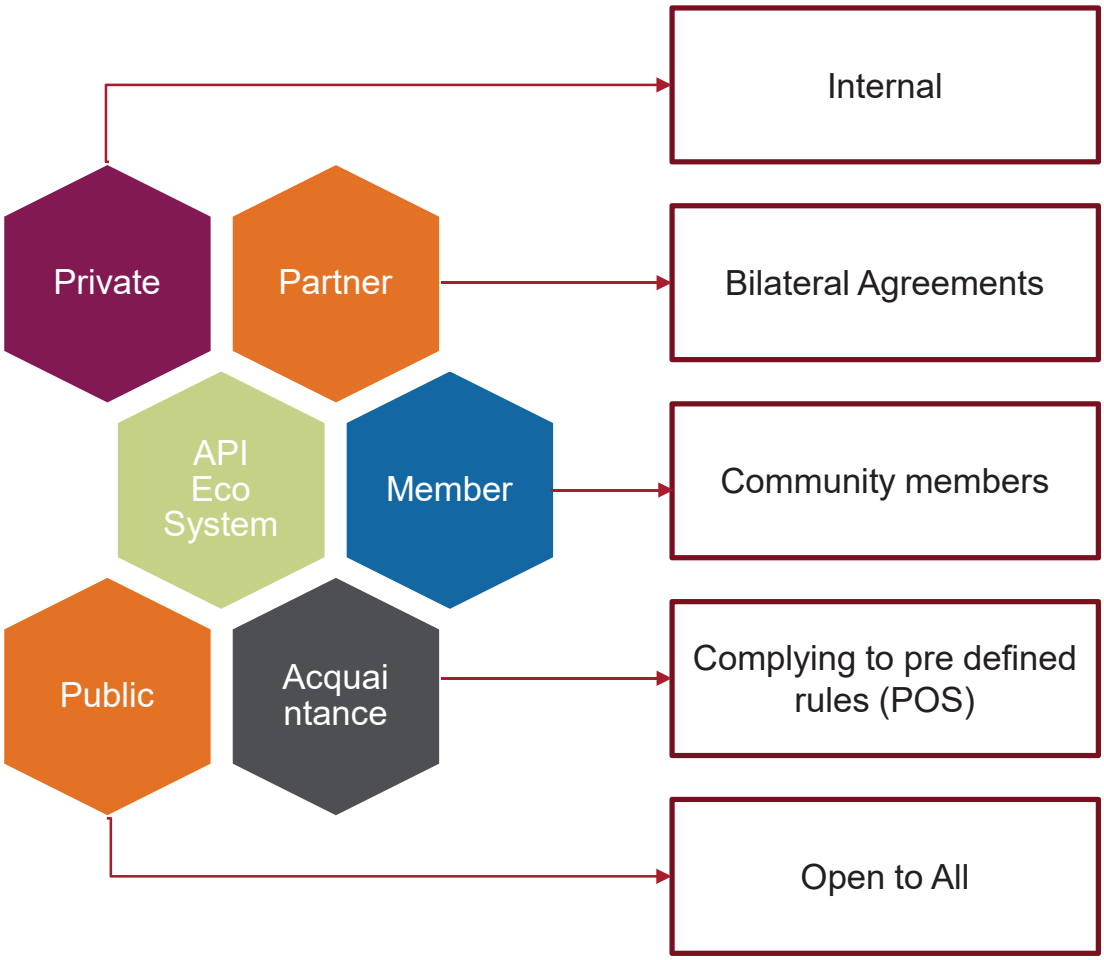
Hence the 2nd law **API Usage API Value * API Ecosystem**

Few terms that need to be introduced:

API ecosystem - The API ecosystem encourages API providers and consumers to collaborate in order to deliver previously unimaginable customer experiences.

API Economy - "The API economy is an enabler for turning a business or organization into a platform." according to Kristin R. Moyer, vice president and distinguished analyst at Gartner. "Platforms multiply value creation because they enable business ecosystems inside and outside of the enterprise to consummate matches among users and facilitate the creation and/or exchange of goods, services and social currency so that all participants are able to capture value."

For the API ecosystem to be built, we first need to recognize the API economy, i.e. identify partners into categories.



After that, you need to build APIs as per the needs of the business participants around their assets and identify the value of those APIs for others in the ecosystem. Post that, you need to bundle the APIs related to similar assets as products and finally package the related API products as API packages in order to target various developer needs.

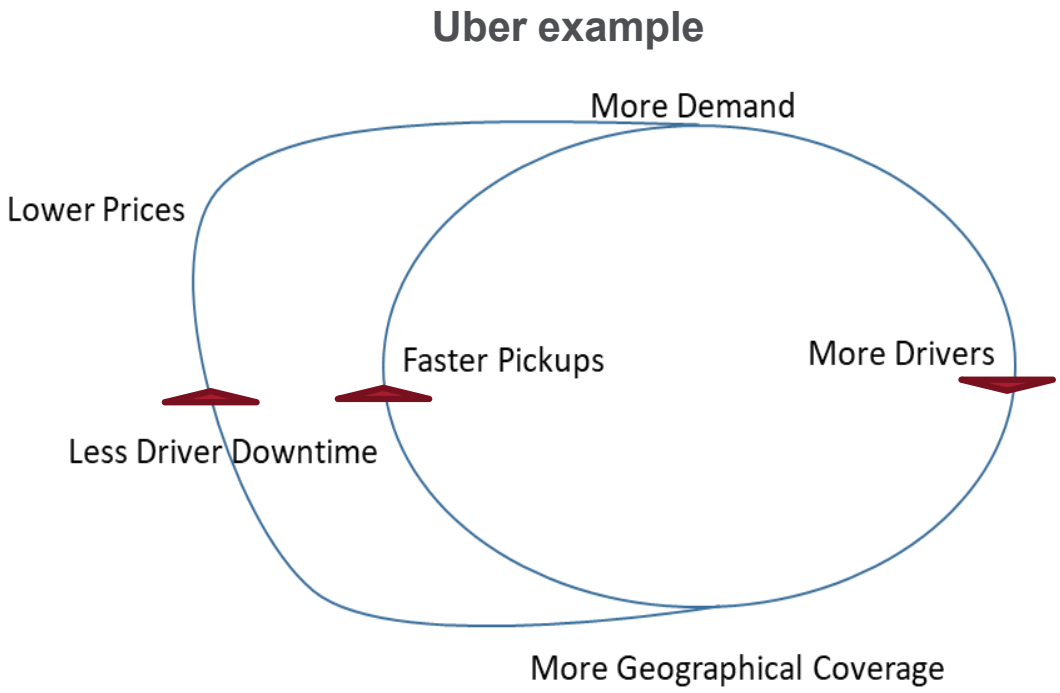
The above steps lead to forming a foundation for a possible successful ecosystem as it facilitates:

- Stakeholder collaboration
- Lowering of barriers
- Internal and external innovation
- Leading to self-service ideas
- Bundling of offerings from partners
- Idea generation which leads to building of new business models

Uber is possibly the best example of an API ecosystem. It is a good example of how API created business use cases where partners and customer all come together to build an ecosystem. Uber is an app-based taxi service provider which does not own taxis and is essentially an aggregator. Its partners are API providers such as Google (Maps), Taxi Owners (Drivers), and payment gateways.

The Uber app helps customers find drivers available nearby who could help them transit between locations. To enable this, Google APIs (Maps) provide the necessary support such as location, distance, etc. related to the customer's request to Uber. Uber, in turn, adds the necessary filters and identifies the driver who could facilitate the transit. The ecosystem further strengthens the fact that Uber's APIs are available for Google Maps – Users of Google Maps can book taxi services right from app, instead of going to the Uber application on their devices.

APIs help connect and bring producers and consumers together to provide a good cab experience. Customers would be happy if the pickup is faster, the geographical coverage is more, and prices are lower, while the drivers would be happy only if their utilization/demand is more. APIs play a key role in maintaining this delicate balance of demand and supply.



Ecosystem impact for Partner

Provide Partners direct access to data, customer bases via APIs

- New revenue growth potential, new business cases
- Increases stickiness with partners
- Increase market share, transaction volumes and revenue

Ecosystem impact for Customer

Provide Customers direct access to logical services via APIs

- Makes easier for customer to transact
- Improves Customer relation, experience
- Increase ease in consuming the services as one logical unit

4.3 Monetization

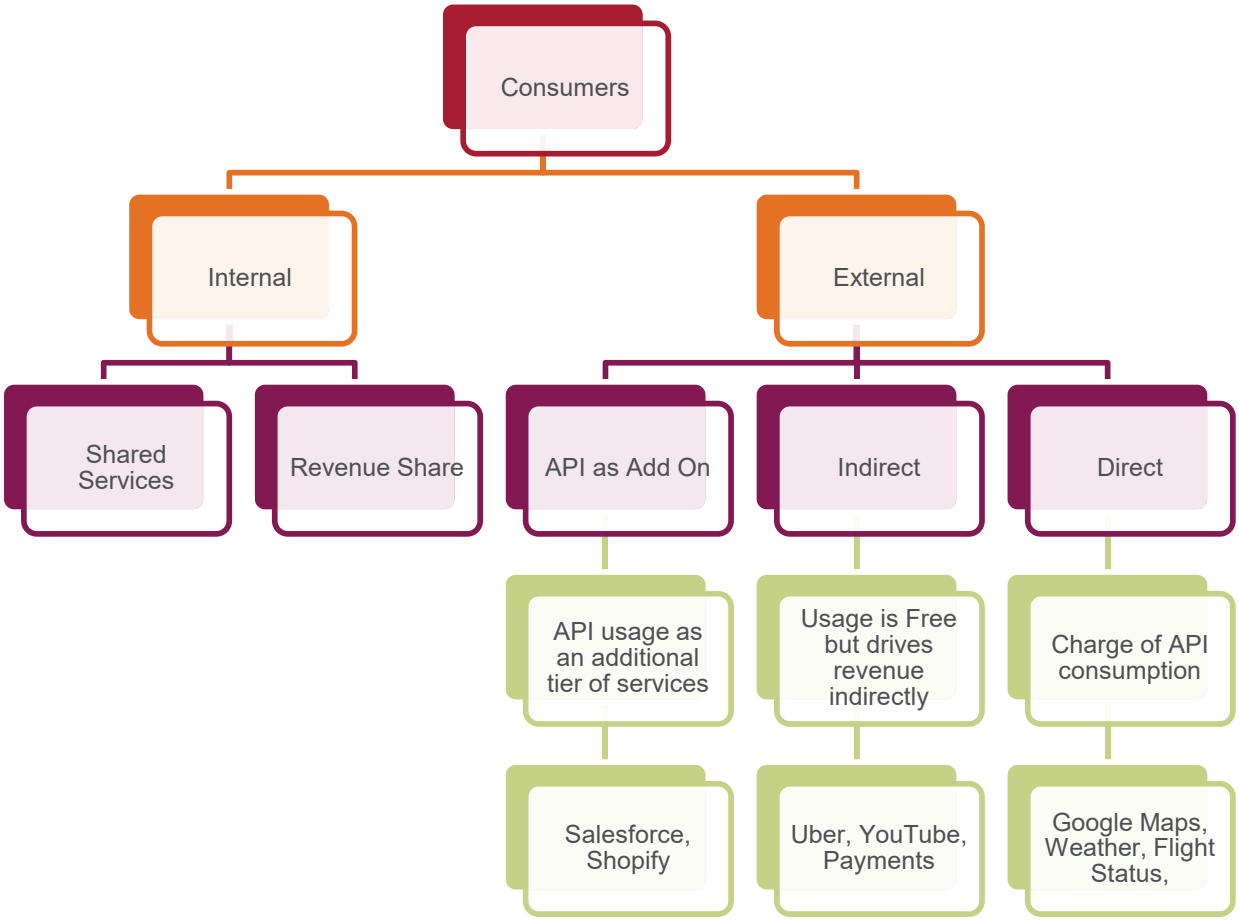
Business is generally about revenue, either direct or indirect. In the context of API, revenue is driven by consumption of APIs, i.e. either the consumption is charged or API consumption drive the organization's business.

In the case of Uber:

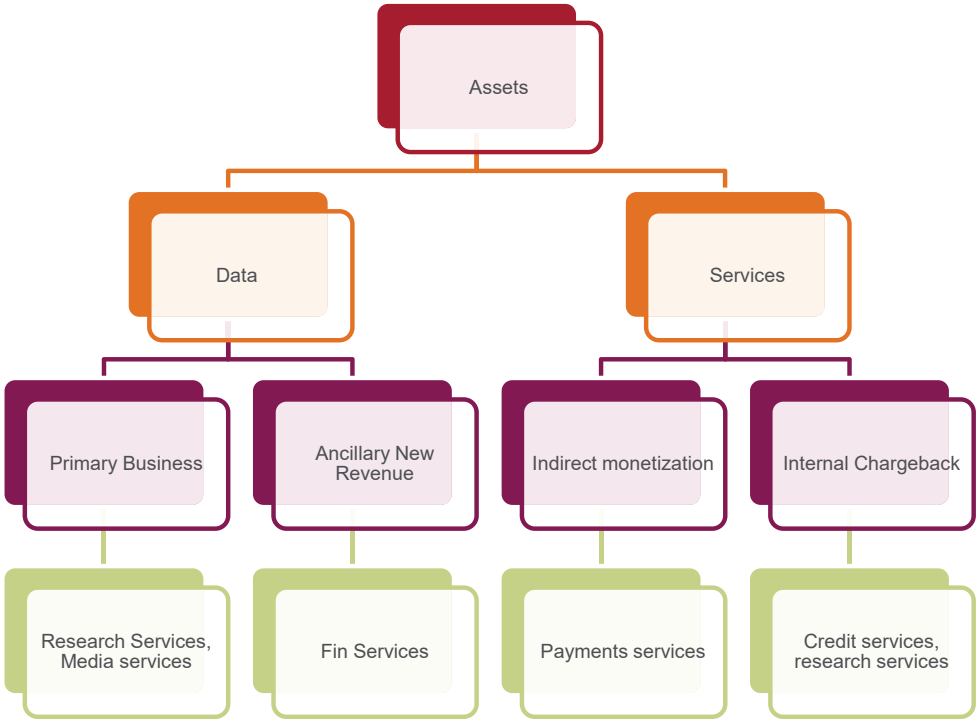
Uber APIs consumed by Google Maps may not be charged, as they drive Uber's business. On the other hand, Google may charge Uber for consuming its APIs for Maps.

Hence the 3rd law **Revenue α (API Value)** $^{\text{API Ecosystem}}$

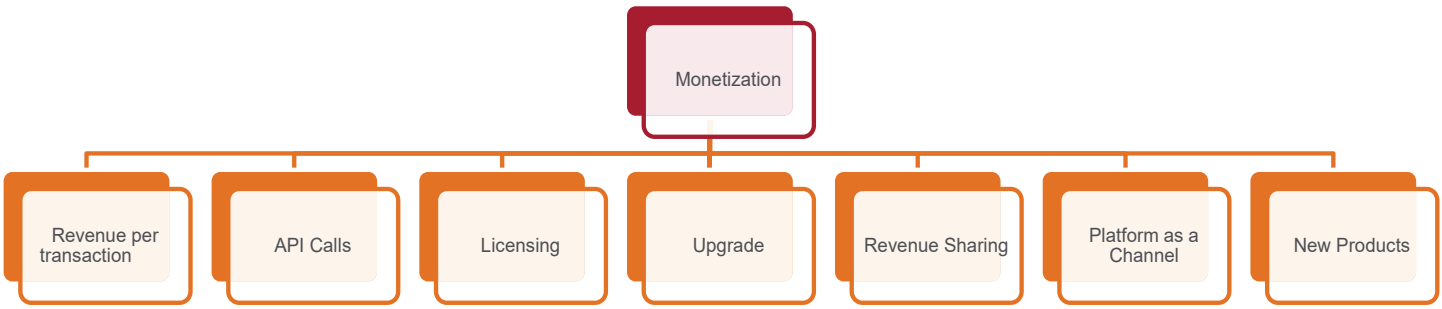
The concept of monetization for an organization can be driven by two aspects, Consumers and Assets. The consumers are generally internal or external for an organization.



Assets for an organization are based on what it can sell or monetize, which is either Data or Services.



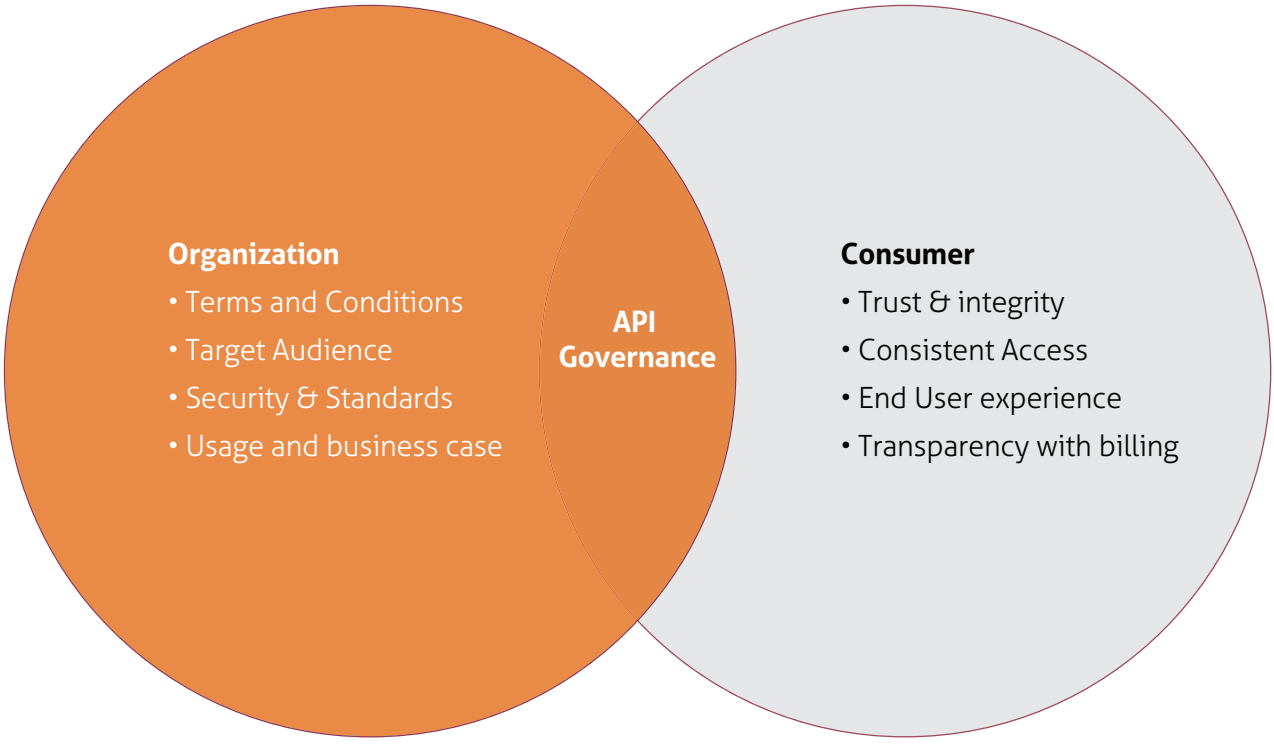
Now, for an organization to monetize its offerings, there needs to be a business model based on which the charging/ revenue could be generated or shared.



4.4 Governance and API team

The governance and API teams are different; the former deals with ensuring standards, security, access control etc., while the latter deals with API implementation - addressing Minimum Viable Product (MVP), API experience and business value.

Governance involves enforcing a set of policies not just in runtime, but also throughout the design and development process. It deals with a lot of moving parts to ensure that the APIs are consistent across the organization. Problems such as duplicate code are prevented, tight coupling between components are ensured and problems such as unreliability, too many services and standards are addressed. The right governance mechanism helps both the organization and the consumer to build trust in the ecosystem.



Assessing API governance requires consideration of the following aspects:

Parameters	Understanding
Centralization	A single source (team) of authority that creates and decides on policies and enforces the same across the organization.
API Contract	Reusability, standardization and consistency starts from here. Establishing a contract helps ensure that APIs are consistent and reusable. It also allows development teams to work in parallel.
Versioning	Businesses keep evolving and so do the needs for an API. Versioning helps developers keep track of and maintain different versions of APIs. The versioning and related notes could help organization take informed decisions on deprecation of API as needed.
Deprecation Policy	Deprecation is a process of removing the earlier version of an API. Versioning, along with good documentation, helps make it very easy for an organization/ developer to know what to expect as APIs are deprecated. Versioning and Deprecation together bring a lot of stability in the API usage.
Automation	Is a large topic which definitely consists of DevOps, but across the SDLC, there are many steps which can be automated such as API contracts, documentation, tracking, code review etc. Organizations can look at multiple tools available today that automate API design and governance processes.
Reusability	Duplication of functionalities is a major concern for organizations. It consumes time, resources and adds overhead during migration, consolidation, enhancements etc. By ensuring the right categorization of services, documentation and discoverability, you can manage the challenges related to reusability.
Style	Establishing and enforcing style guidelines for all APIs ensures readability and consistency.
Tracking	Governance can't be enforced without tracking. Organizations should know where the APIs are deployed, how they are being used, who and how many consumers are there for the APIs. This helps in taking informed decisions for scaling, migrating, downtime etc.
API Discovery	Once the organization wants to drive self-service after creation of the APIs, the ability to search and discover the latter is critical from the user experience perspective. A method for depicting dependencies also becomes necessary for analyzing the future impact of apps consuming your APIs.

Parameters	Understanding
Security	This involves establishing who can access each API (business asset) and setting up the visibility and security enforcement around the API. The decisions regarding API security needs to be in alignment with overall organizational security governance policies.
Compliance	Compliance ensures APIs and the platform are in order by implementing and adhering to standards and best practices.
Monetization	This involves deciding the business model and setting up the monetization capability with the right measurements and policies for enforcements.
Ownership of Data and Services	This involves establishing governance and processes to define ownership of data and services common across LOBs. This ensures reduction of multiple versions of APIs.

There are other factors that can also be considered such as Change management, Auditing etc., but we would like to assume that they would be consistent across the organization.

The API team's responsibility cuts across the entire API lifecycle. It is important to first understand the responsibilities of the team and identify roles which suit them. The focus is not on enforcing a team structure that is centralized, federated or distributed, but on allocating the responsibilities to the roles identified such that the organization is able to deliver API products to developers consistently and eventually support the business.

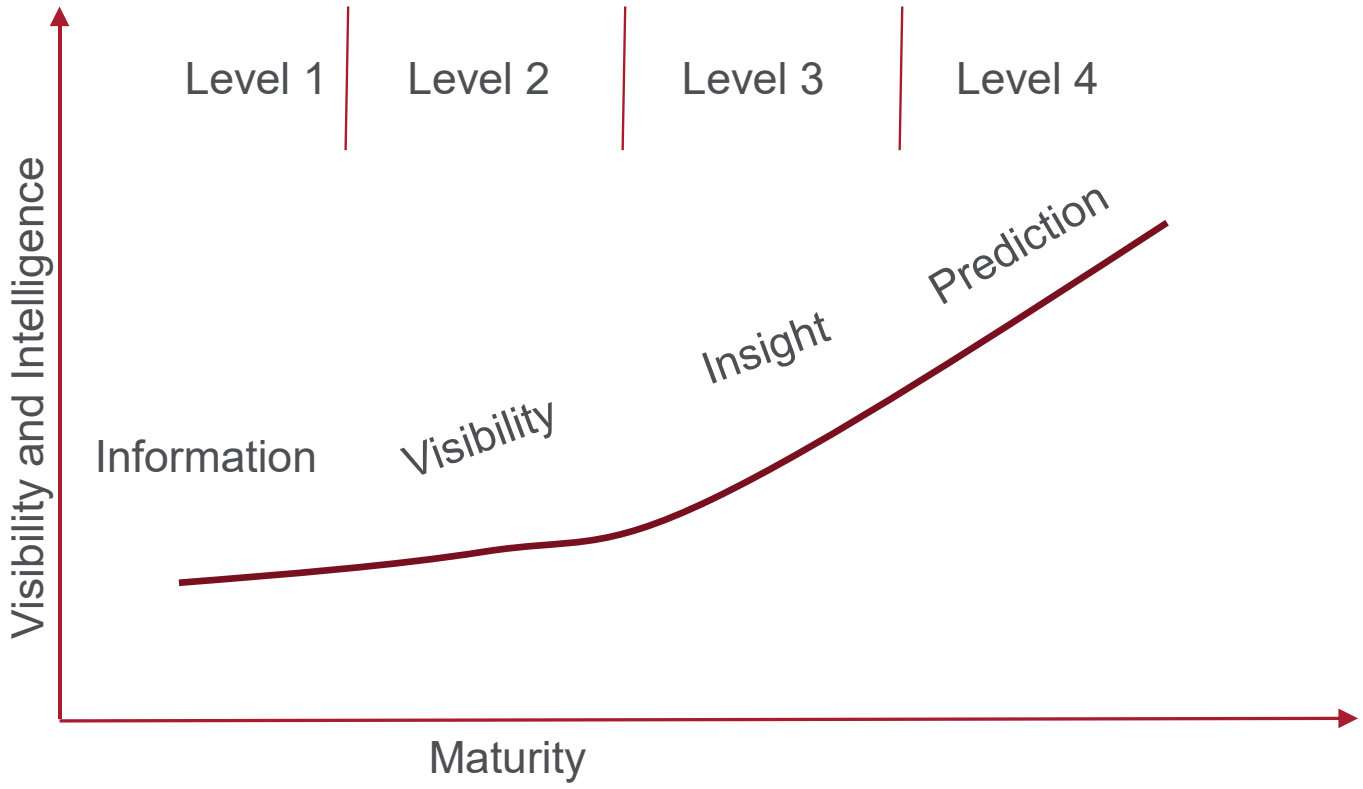
The team's main objective is to bring uniformity in the API creation process by ensuring that APIs are created for consumption and that access to applications and other back-end data is provided in a consistent manner. Having said that, the expectation of the organization does not end there. It would need the API team to think in terms of what business value they are driving, the developer experience are they providing and finally, the product they are building.

The expectations from API team leads to the following roles:

Expectation	Role	Responsibility
API product, Developer experience	API Architect	Responsible for planning, designing, and reviewing the API builds. They ensure the standards and guidelines are met as defined by the governance team. They are the thread which ties the requirements, existing implementations (backend), NFRs, guidelines, needs of the developers consuming the APIs and eventually designing APIs that meet the needs.
API product, Developer experience	API Developer	The architect provides the skeleton for the API through designs, while the developer needs to fill in with the code and related documentation, which would help o ther developers consume the APIs easily.
Developer experience	API Evangelist	API evangelist is a very critical role and is responsible for ensuring that the APIs meet their end objective of being consumed. This role is responsible for managing the developer portal and ensuring developers have access to detailed information on the product offering, including documentation etc. The evangelist also needs to bring awareness of the APIs to communities within and outside of the organization through marketing campaigns, events, road shows discussions etc. as a way to expand the ecosystem.
Business value	API Champion	They are the drivers and influencers from the business side and capable of converging an organization's API programs to align with corporate goals. They would also develop strategic business goals linked to the API product offerings that benefits the organization.
Overall	API Product Manager	API product manager is the SME for the API. They are responsible for converting organization's vision for Digital Service/ Offerings to API products. They are r esponsible for managing the API product lifecycle right from the requirements stage to the implementation. As the SMEs, they understand the technology and are responsible for creating a roadmap for achieving the expected business value. They are the go-to people in the organization for coordination between different stakeholders and prioritizing activities as part of the API roadmap.

4.5 API Monitoring

API monitoring refers to the practice of monitoring APIs, most commonly in production, to gain visibility into performance, availability and functional correctness. API monitoring capability is designed to help organizations analyze the performance of applications and take informed decisions to improve poorly performing APIs. They provide a measure of how long a routine takes to execute, how often it is called, where it is called from, and the total time spent in executing that transaction etc. IT organizations require a deeper and more precise understanding of what is happening across their IT environments. According to a survey by Enterprise Management Associates (EMA), most enterprises find it difficult to find the right monitoring strategy to manage their environments. At the same time, over 65% of enterprise organizations have more than 10 monitoring tools. These monitoring tools could be home-grown and specific to a project or task. Now, the level of API monitoring maturity depends on what an organization wants to gauge. A holistic approach for API monitoring not only helps detect, diagnose and address performance issues, but also helps organizations trust their systems and understand the various bottlenecks and impacts on the platform. Based on the level of understanding, there are typically four levels of monitoring.

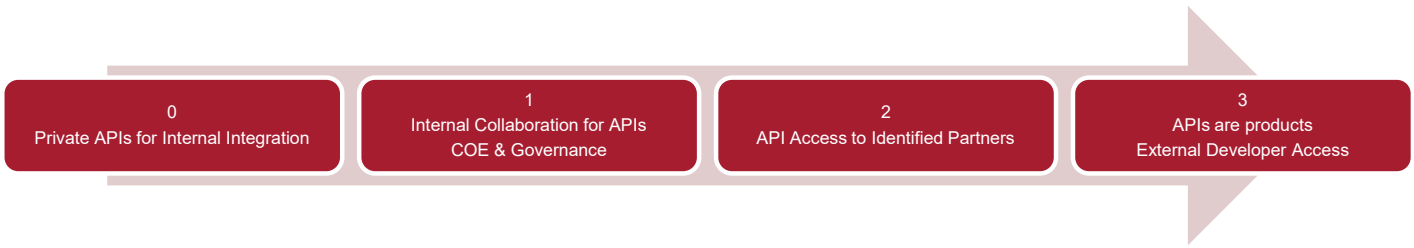


Level Names	Details
Information	At this level, organizations possess basic capabilities for API monitoring. They have the capabilities to understand health of the APIs, get notifications. However, they would not understand why they are underperforming, in case they are.

Level Names	Details
Visibility	At this level, organizations will capitalize on available information through monitoring capabilities which provide metrics, logs, and traces. The information would be helpful for tracing root causes, if needed. The visibility would still be at a level of the API and will not provide the information (status/ availability) of its dependencies such as databases, FTP servers, file server, topics and queues in messaging layer etc.
Insight	This stage is about what most organizations would aspire to be at a minimum. Organizations have a consolidated view of APIs along with their dependencies which helps them: • Reduce time to assess impact due to down time • Reduce time between issue reporting and issue resolution Though organizations have the capability to sense and detect, their actions would still be reactive.
Prediction	Prevention is better than cure. The capability of the organization at this level helps them access insights which helps them take proactive measures before failures are reported. The solutions could involve scaling of APIs for load predicted, stopping APIs if needed, and taking measures for throttling the requests and beyond.

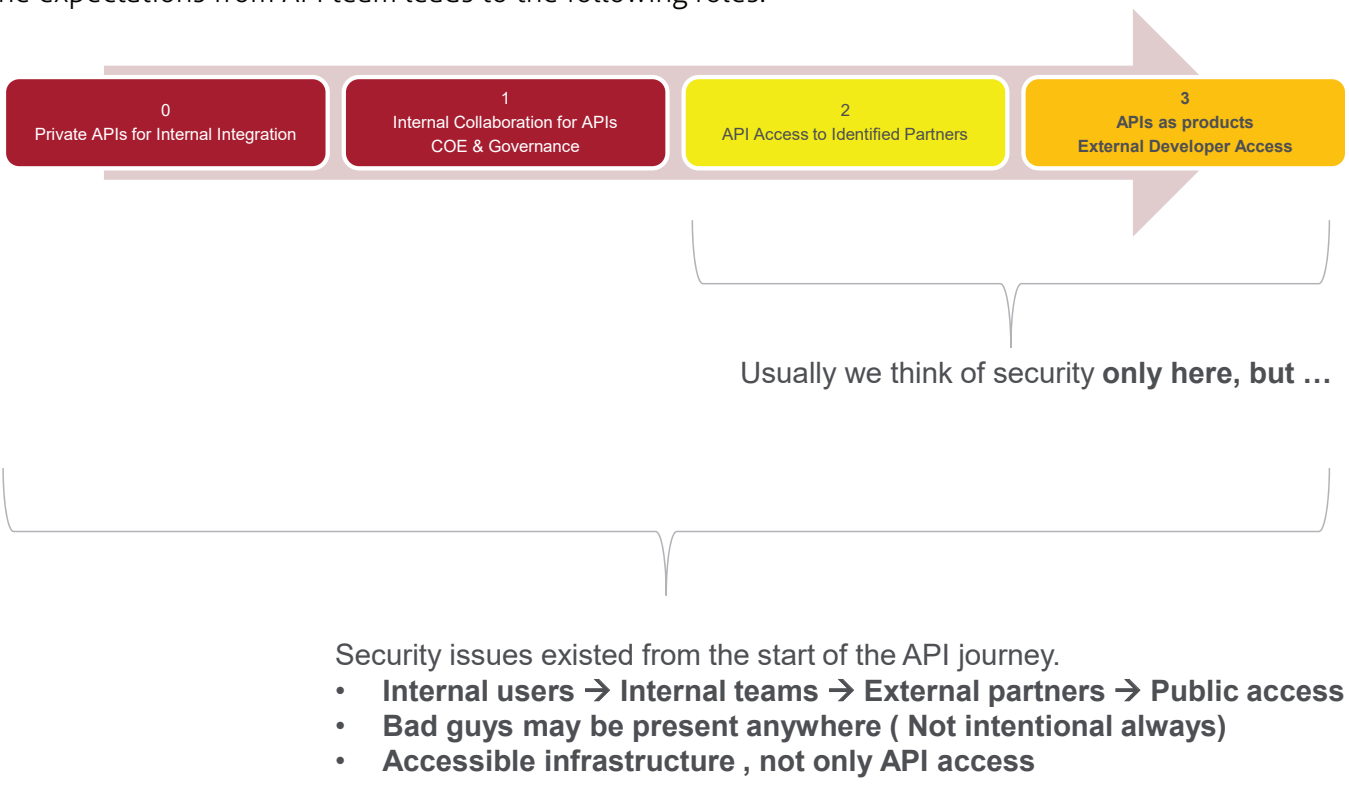
4.6 API Security

Security is crucial in any implementation. When an organization decides to implement API, it must look at the opportunity holistically and ensure that the security is catering to all needs of applications, data, and infrastructure. Before the organization gets to the next level, it is necessary to understand where to start.



Where does Security kicks in?

The expectations from API team leads to the following roles:



There are multiple facets of API security and it is the responsibility of the implementers to understand the current implementation, future needs, identify the gaps and address them across

- Authentication & Authorization
- Data (Encryption/Decryption)
- Monitoring
- Infrastructure

5. Conclusion

In the digital era, where businesses are aggressively selling their offerings and nurturing an ecosystem, sitting on the sidelines is no longer an option. Organizations need to ensure that the customer experience is optimal, and their services, data and offerings are available digitally to the consumer for usage. Obvious steps include an organization's need to understand where they are on the API journey and its maturity. Based on the various maturity parameters, capabilities, and possibilities, we have created an API maturity model that can provide organizations the means to assess their API program maturity.

6. Reference

https://cloud.google.com/apigee/resources/ebook/api-first-security-dont-build-your-own-maginot-line-confirmation-ty
https://www.gartner.com/smarterwithgartner/welcome-to-the-api-economy/
https://www.eginnovations.com/white-papers/Overcoming-IT-Monitoring-Tool-Sprawl-with-a-Single-Pane-of-Glass.pdf

7. About the Authors

Vijay Chakka possesses close to 17+ years of IT industry experience, primarily in the integration space, and is engaged in consulting, managing delivery and building practice. Currently, he heads the 'MuleSoft CoE' within the Enterprise Application Integration service line under Mindtree Digital. He has been part of the growth path for multiple organizations across industries such as banking, entertainment, energy, manufacturing etc.

Surendra Thekkatte is a CoE head within Enterprise Application Integration service line under Mindtree's Digital practice with over 22 years. He is specializes in integration and architecture consulting offerings, helping customers defining their API strategy, setting up API CoEs and implementing the roadmap, transitioning from SOA-centric middleware products to microservices-based architecture involving API management platforms.

Gopalakrishnan Srinivasan has 18+ years of experience in IT and has been involved in many phases of the software development life cycle on Enterprise Integration Application and Java/J2EE-based projects, apart from working on various RFP/RFIs. He is well-versed with Middleware technology of MuleSoft and Java/J2EE. He has 9+ years of experience in Mule ESB product suite (version 2.x, 3.x & 4.x) and has been a part of MuleSoft ESB Center of Excellence for 4+ years in many organizations. Gopalakrishnan has worked on various business domain such as Energy & Utilities, BFS, Insurance, Education, Retail, Media, Life science and Healthcare.

About Mindtree

Mindtree [NSE: MINDTREE] is a global technology consulting and services company, helping enterprises marry scale with agility to achieve competitive advantage. "Born digital," in 1999 and now a Larsen & Toubro Group Company, Mindtree applies its deep domain knowledge to 300+ enterprise client engagements to break down silos, make sense of digital complexity and bring new initiatives to market faster. We enable IT to move at the speed of business, leveraging emerging technologies and the efficiencies of Continuous Delivery to spur business innovation. Operating in 18 countries and over 40 offices across the world, we're consistently regarded as one of the best places to work, embodied every day by our winning culture made up of over 21,000 entrepreneurial, collaborative and dedicated "Mindtree Minds."