Hype Cycle for APIs, 2023

Published 25 July 2023 - ID G00792664 - 95 min read

By Analyst(s): Mark O'Neill, John Santoro

Initiatives: Software Engineering Technologies; Adopt Modern Architectures and Technologies; Build and Deliver New Digital Products/Experiences to Drive Business Results

The role of APIs in software engineering is changing, with innovations including API observability and federated API gateways. Gartner's 2023 Hype Cycle for APIs highlights API technologies and practices that are key to success for software engineering leaders and their teams.

Analysis

What You Need to Know

API technologies and skills are more important than ever. Gartner's Software Engineering Leaders Role Survey, published in January 2023, found that API design and API management skills are high in demand, second only to application security skills. The two top two skills in demand — API skills and security skills — demonstrate the importance of API security.

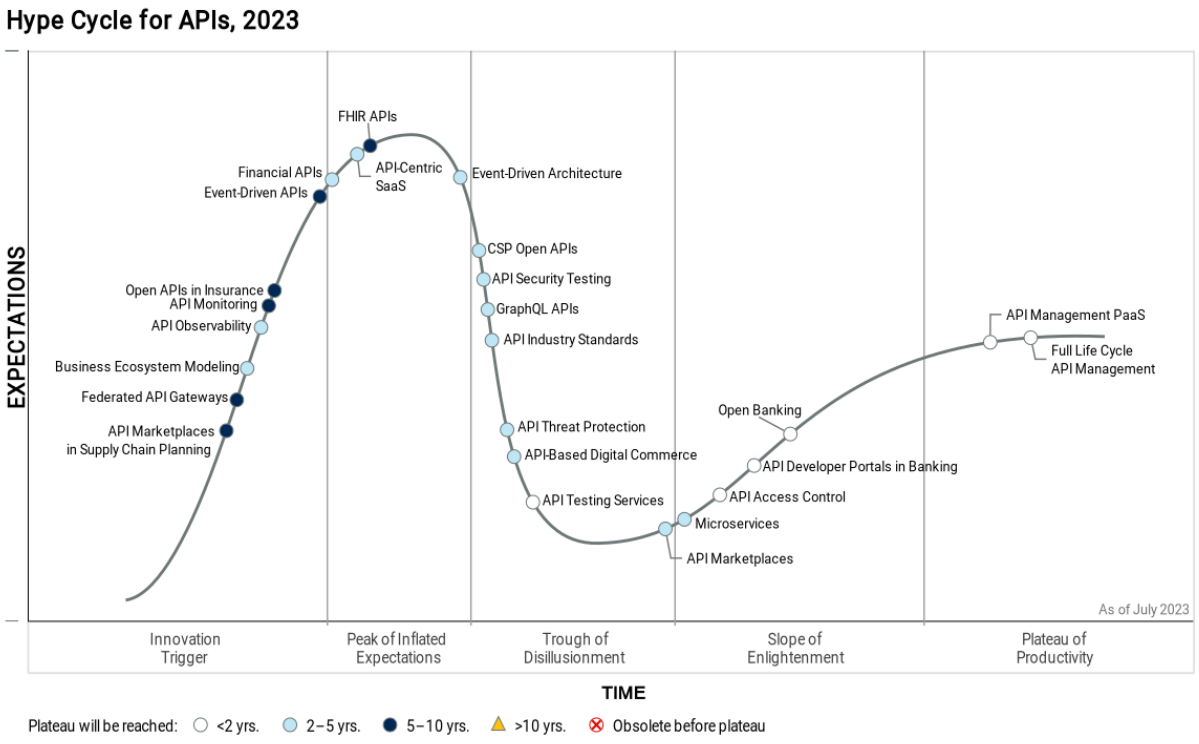
At the same time, the personas involved in API management are changing. Developers have become more central to product selection. In Gartner client inquiries, we hear that organizations increasingly require API management solutions that are developer-friendly. This means making APIs available for tasks such as registering APIs in the API management solution itself. This reflects the growing influence of software developers overall, who value automation, including the use of GitOps flows in API deployment.

The Hype Cycle

This year's Hype Cycle for APIs highlights the most important trends, including:

- Federated API gateways are growing more popular. As API usage continues to grow, organizations must manage multiple API initiatives, involving multiple cloud environments and multiple teams. This leads to a requirement for API management solutions that span multiple API gateways. For more information, see Reference Model for API Management Solutions.
- API security continues to be important. API security testing, API access control, and API threat protection have increased in maturity in this year's Hype Cycle for APIs. This reflects the continued need for API security, as well as the maturation of API security tooling and practices. For more information, see Research Index: Everything You Should Do to Address API Security.
- Industry APIs are moving forward. This year's Hype Cycle accounts for communication service provider (that is, telecom) APIs, financial APIs, open banking APIs, and the use of API marketplaces in supply chain. Open banking is ahead of other industries due to the presence of open standards.
- API observability represents a new entry in this year's Hype Cycle for APIs. This reflects the importance of API dependency analysis and usage monitoring. API observability solutions enable developers and API owners to see the impact of API changes, including detecting breaking changes before they happen.
- Both API management PaaS and full life cycle API management are now on the Plateau of Productivity. This is because they both are mature technologies with many customer deployments.

Figure 1. Hype Cycle for APIs, 2023



The Priority Matrix

Open banking is identified as a transformational technology with near-term maturity. This technology has emerged from the Trough of Disillusionment — a period in which many banks reported disappointment with their open banking initiatives. The emergence of PSD3 (Payment Services Directive 3) from the European Union, as well as other mandates worldwide, has moved open banking forward and out of the Trough of Disillusionment. For more information on open banking use cases, see Emerging Use Cases That Validate the Business Value of Open Banking.

Other industry vertical APIs are further from mainstream adoption. This includes communication service provider (CSP) open APIs, insurance APIs, Fast Healthcare Interoperability Resources (FHIR), financial APIs, and API marketplaces in supply chain planning.

CSP open APIs are identified as transformation, as is API-based digital commerce, which includes “headless” commerce. However, at two to five years, the path to mainstream adoption is longer than open banking.

No technologies have a time to maturity of over ten years, which shows the fast-moving nature of the API world.

Table 1: Priority Matrix for Hype Cycle for APIs

(Enlarged table in Appendix)

Benefit	*Years to Mainstream Adoption*			
↓	*Less Than 2 Years* ↓	*2 - 5 Years* ↓	*5 - 10 Years* ↓	*More Than 10 Years* ↓
Transformational	Open Banking	API-Based Digital Commerce CSP Open APIs		
High	API Access Control API Testing Services Full Life Cycle API Management	API-Centric SaaS API Observability API Security Testing API Threat Protection Business Ecosystem Modeling Event-Driven Architecture Microservices	API Monitoring Federated API Gateways FHIR APIs Open APIs in Insurance	
Moderate	API Developer Portals in Banking API Management PaaS	API Industry Standards API Marketplaces Financial APIs GraphQL APIs	Event-Driven APIs	
Low			API Marketplaces in Supply Chain Planning	

Source: Gartner (July 2023)

Off the Hype Cycle

On the Rise

API Marketplaces in Supply Chain Planning

Analysis By: Amber Salley

Benefit Rating: Low

Market Penetration: Less than 1% of target audience

Maturity: Emerging

Definition:

An API marketplace is a platform to share APIs. They range from basic API catalogs, to API developer portals from a single API provider, to commercial marketplaces with APIs from many providers. Consumers, mainly developers, (both vendors and end users) use API marketplaces to discover APIs and, in some cases, may purchase access. Although public API marketplaces are better-known, a growing number of organizations are deploying internal or private API marketplaces.

Why This Is Important

API marketplaces enable organizations to publicize their APIs. They are usually associated with external marketplaces that share APIs with a community of developers and enable partners to implement solutions using the APIs. However, as most APIs are meant for consumption by teams within an organization, marketplaces can also be internal. They make it easier to find APIs internally, helping with wider sharing of capabilities between different business units, product and development teams.

Business Impact

API marketplaces within supply chain planning (SCP) solution environments enable users to extend the capabilities of their core planning platform. An API marketplace can increase developer visibility and user mind share, drive API usage, and by extension, increase business impact. API consumers can use marketplaces to simplify finding and comparing different APIs when they are looking for specific functionality but have not selected exactly which API to use.

Drivers

- The number of APIs within an organization is climbing, driving the need for developers to more easily discover which APIs and services are available.

- Composable business, including composable supply chain planning, relies on the use of API marketplaces to share APIs and packaged business capabilities.
- Increased use of low-code platforms, integration platforms, robotic process automation (RPA) and analytics tooling enables more citizen development, using APIs that may be sourced from API marketplaces.
- New open-source platforms, such as Backstage from Spotify, are driving the creation of internal API marketplaces as part of larger developer hubs.

Obstacles

- There are very few supply chain planning software vendors that provide any environment resembling an API marketplace. Some are building partner ecosystems, but few have shared a vision to establish a public API marketplace.
- Public API marketplaces that provide a public directory of APIs from multiple providers have generally had disappointing results, as developers are more likely to go directly to API providers to sign up for APIs. Therefore, SCP vendors may be hesitant to attempt to launch a public API marketplace. However, internal API marketplaces have had more success for API providers since they enable developers to share APIs across multiple teams. Time will tell if the same holds true for SCP API providers.
- API portals provided as part of API management platforms are typically basic in nature, resulting in significant customization work to create a customer-oriented API marketplace.

User Recommendations

SCP API providers should:

- Create an internal API marketplace focused on the needs of software engineers to share APIs across the organization, as part of an internal developer portal.
- Manage senior business stakeholders' expectations by ensuring they are aware that outcomes from placing APIs in public API marketplaces are often disappointing.
- Examine billing terms to understand what goes to the marketplace provider when considering commercial API marketplaces.

- Establish a commercial model upfront (e.g., through registration fees and/or revenue share) and a clear governance process for onboarding third-party APIs if you plan to build your own API marketplace.

SCP API consumers should:

- Ensure that you use APIs from trusted marketplaces and trusted API providers, examining usage agreements, licensing and billing terms carefully.
- Investigate whether consuming an API directly from the API provider offers better pricing or usage terms than consuming the API via a marketplace.

Sample Vendors

Achieve Internet; Bump; Postman; Pronovix; ReadMe; SmartBear (Swagger); Spotify (Backstage); Stoplight

Gartner Recommended Reading

Innovation Insight for Internal Developer Portals

Reference Model for API Management Solutions

Federated API Gateways

Analysis By: Paul Dumas

Benefit Rating: High

Market Penetration: Less than 1% of target audience

Maturity: Emerging

Definition:

Federated API gateways are data planes of heterogeneous API gateways managed by a common control plane. The control plane enables central governance of the data planes and supports the automation of API gateway policy changes. To satisfy specific use cases, the individual API gateways will vary in deployment model, capability set and scope of responsibility.

Why This Is Important

Organizations need to be able to use different API gateway technologies in their architectures in different circumstances. However, the challenge lies in how to implement central administration of diverse gateways. Federated API gateways enable the management of multicloud and multivendor API gateways for scalability, consistency, security and a unified experience for API consumers and providers.

Business Impact

Federation coordinates multivendor API gateways. This enables businesses to gain the advantages of native API technologies in multicloud and on-premises environments, while minimizing the complexity of governance and administration. Federated API gateways help businesses comply with industry regulations regarding data location and API standards, have an agile architecture for innovation, and better assimilate systems acquired through mergers and acquisitions.

Drivers

- Enterprises utilize multiple cloud-native API gateways, as well as API gateways from open-source and commercial sources.
- Enterprises use varying API gateways to address different implementation models and satisfy specific use cases. For example, enterprise and departmental gateways provide extensive, coarse-grained capabilities required to handle external traffic and serve applications. These gateways are equipped to process high volumes of traffic and large sets of APIs. Behind enterprise API gateways, developers prefer lightweight gateways to front smaller sets of APIs or APIs hosted in different environments. These are low-footprint and may be declarative to support ease of deployment and scaling. Site reliability engineers find lightweight API gateways compelling because of their efficiency and operational simplicity.
- Organizations using a zero-trust security posture rely on a hierarchy of API gateways to accomplish security measures in the flow of orchestrated APIs.
- The API gateway landscape continues to expand its offerings, as gateway solutions battle to be the preferred option for one or more architectural models and API use cases.

Obstacles

- API gateway technologies do not all have parity with mediation capabilities, and there is no commonly expected base set of mediation features.
- There is a marked lack of interest among vendors to make their API gateways compatible with other vendors' control planes.
- A lack of policy standardization causes vendors to have independent opinions of gateway policies and how they are defined.
- Security mechanisms and capabilities vary across API gateways.

User Recommendations

- Many API gateway vendors offer variants that can be deployed to multiple cloud environments and on-premises. Look for solutions that maximize same-vendor gateways in your architecture while still meeting your requirements.
- Clearly identify API policies that are to be consistent across the enterprise. Ideally, policies related to security, authentication, usage and traffic management should be targeted for enforcement in API gateways. Protocol and format mediation (SOAP to REST, XML to JSON) and simple data marshaling and orchestration are also seen in API gateways, but avoid including complex processes and business logic.
- Support centralizing API monitoring and analytics across heterogenous API gateways by using solutions that capture and aggregate gateway logs and API usage metrics. OpenTelemetry solutions are also becoming more prevalent.
- As API gateways become commoditized, look for industry consensus on standard sets of policies for them.

Sample Vendors

APIwiz; Postman; SmartBear

Business Ecosystem Modeling

Analysis By: Ian Reynolds, Auria Asadsangabi

Benefit Rating: High

Market Penetration: 1% to 5% of target audience

Maturity: Emerging

Definition:

A business ecosystem model is a dynamic network of entities that interact to co-create and exchange sustainable value for participants. Business ecosystem modeling focuses on extending the scope of business architecture to the customers, suppliers, devices, partners and other entities that make up an organization's ecosystem.

Why This Is Important

All organizations exist in business ecosystems that include customers, partners, competitors, regulators, suppliers and other entities. The business ecosystem is made up of a complex set of relationships, roles and dynamics. Ecosystem modeling can provide insight into a business ecosystem and its dynamics, and aid in developing effective business strategies.

Business Impact

Although organizations have always existed in business ecosystems, the digital age has accelerated the complexity and the number of connecting relationships between participants. Business ecosystems now extend around the globe, mediated by technology, and many business models are based on business ecosystems. Business ecosystem modeling enables organizations to better understand and operate within their ecosystem.

Drivers

- Business strategy in a digital era increasingly relies on adopting an ecosystem mindset to ensure organizations are maximizing value realization from ecosystem opportunities.
- Business ecosystems are, by nature, complex adaptive systems. Effectively modeling them enables decision makers to play out scenarios and shape more-sophisticated strategies.
- Business ecosystems facilitate trust-based partnerships and enable open innovation — using the resources of partner organizations and building them into the organization's business and operating model.
- Advancements in technology have helped forge new, and develop existing, interconnections between organizations and their macro environment. Ecosystem modeling helps organizations understand the nature and dynamics of these interconnections.

Obstacles

- Business ecosystems represent a substantial change in perspective, away from zero-sum thinking toward a positive-sum perspective.
- New skills, competencies and tools are needed to model and understand the dynamics of the business ecosystem. This is particularly important as the volume of inputs into the model increases and makes understanding ecosystems more complex.
- Data science, simulation and statistical analysis are all complementary skills for more advanced ecosystem modeling, but they are in short supply.
- Some modeling tools are available, but they are not fully mature and lack widespread adoption and understanding.

User Recommendations

- Begin by learning how organizations have used business ecosystem modeling to optimize and transform their operations.
- Use business ecosystem modeling to identify the participants, their roles, their relationships and their interrelationships. A business ecosystem model can highlight monetization opportunities, threats and challenges.
- Foster a mindset shift among enterprise leaders, promoting openness toward modeling and engaging with the organization's business ecosystem.

Sample Vendors

Avolution; Bizzdesign; Inlecom; Tr3Dent; WorkSpan

Gartner Recommended Reading

Model Your Ecosystem to Identify the Partners Needed for Digital Business

EA's Evolving Role in Digital Business Ecosystems: Benchmark Data

Case Study: An Ecosystem Lens Optimizes End-to-End Customer Journeys (Premera Blue Cross)

Case Study: Methods to Build Trust-Based Ecosystem Partnerships to Optimize Business (RubyMeadow*)

Case Study: Ecosystem Modeling Workshops to Develop Partnerships (Standard Bank Group)

API Monitoring

Analysis By: Paul Dumas

Benefit Rating: High

Market Penetration: 1% to 5% of target audience

Maturity: Emerging

Definition:

API monitoring is the practice of assessing the availability and performance of APIs. It provides real-time tracking of the service level that API consumers are experiencing, and can trigger alerts when intervention is needed. API monitoring also includes API traffic analysis for API usage insights and provides a security measure by detecting suspicious patterns. API monitoring is used in preproduction environments to identify performance issues early in the API development process.

Why This Is Important

The performance and reliability of APIs directly impacts the performance and reliability of an organization's systems. It is imperative to monitor the state of health for APIs, and to have tools probing for issues that can produce actionable alerts to resolve events. In addition, API analytics provide visibility into API usage patterns to reveal opportunities for improvement, dependency relationships and suspicious behavior.

Business Impact

- Business outcomes depend on systems with healthy APIs; API monitoring identifies anomalies and alerts about distressed APIs.
- Businesses increase API quality and reduce developer rework by using API monitoring during API development to confirm that APIs meet nonfunctional requirements.
- API monitoring confirms compliance with any SLAs, regulations and other runtime obligations.
- API analytics give API product managers visibility to measure the value of APIs.

Drivers

- APIs often invoke an orchestration of subsequent APIs. Monitoring each API in a workflow gives organizations needed insight about API dependencies and expedites troubleshooting efforts.
- Enterprise applications have a growing reliance on both provider-built and third-party APIs. For example, if a third-party authentication API becomes unavailable, then users of websites or applications that use that API will be unable to log in. The need for API monitoring has extended to include third-party APIs to ensure business systems are secure and healthy.
- Similar to third-party APIs from external sources, separate internal teams inside the organization share, and need to monitor, APIs they did not build.
- Public-facing products and ecosystems incorporate externalized APIs that are required to meet SLAs. API-centric (headless) SaaS is a fast-growing business model. Self-service API portals allow new API users to access APIs and increase API request volumes at any moment. Externalized APIs expose surface areas for malicious attacks. These factors have made external API monitoring more complex, but also necessary to maintain trust with customers and partners.
- Because APIs are pervasive across systems including integration, AI, applications, products, RPA and composable platforms, API analytics provide observability into workflows and user behavior. API analysis is a valuable practice for generating insights into process improvement opportunities and new product innovations.

Obstacles

- Architectures with diverse API technologies are both obstacles and drivers for API monitoring. Aggregating across all APIs of an enterprise can present challenges in collecting and merging API usage data into a central monitoring tool. At the same time, multiple API gateways and API implementations create more urgency for API monitoring that provides unified visibility.
- Not all monitoring tools offer machine learning (ML)-based analytics. Some lack capabilities such as anomaly detection, which prevents valuable insight into API usage patterns, API consumer behavior, API performance analysis and related trends.
- Operations teams need central monitoring of all APIs. However, API monitoring tools also need functionality to filter and present select APIs to API product managers who need API analytics to manage their products.
- Cloud-based API monitoring products are not suited to monitoring internal APIs without the addition of internal agents.

User Recommendations

- Organizations with heterogeneous API gateways must evaluate dedicated API monitoring technologies capable of aggregating across multiple API gateway vendors.
- Create opportunities to select advanced monitoring and analytics technologies by implementing agents that capture usage data (or extract logs) from API gateways and then port it to a central tool.
- Prioritize API gateways that work with OpenTelemetry Collector to take advantage of the tools adopting the emerging OpenTelemetry standard.
- Use API monitoring tools that include ML capabilities to provide your organization with in-depth and advanced analytics.
- Confirm the alerting capabilities of your API monitoring tool satisfies your requirements.
- Conduct stress tests of your API monitoring solution to confirm that the burden of harvesting data from the API gateway(s) does not put API SLAs at risk.
- Use API monitoring tools that can leverage synthetic data to simulate load testing and preproduction testing.

Sample Vendors

APImetrics; Moesif; SmartBear

API Observability

Analysis By: Dave Micko

Benefit Rating: High

Market Penetration: 20% to 50% of target audience

Maturity: Early mainstream

Definition:

APIs continue to become more important as software architecture patterns and critical integration tools, so API providers require increasing visibility into how their APIs operate and change. Observation based on telemetry emitted from logs, performance statistics and external monitoring is aggregated and normalized via OpenTelemetry (OTel) standards to provide running snapshots of API performance and reliability.

Why This Is Important

Monitoring business-critical APIs is integral to business operations. From detecting changes to monitoring running states, observability captures data from each phase. API observability has moved from simple log analysis to sophisticated dashboards aggregating telemetry across many APIs, as well as integration with code repositories and integrated development environments (IDEs). Developers and API platform and platform engineering teams are affected by API observability systems and practices.

Business Impact

API observability has grown into dashboards aggregating telemetry into action-oriented analysis. API observability can affect everything from reducing mean time to recover (MTTR) to understanding API dependencies across suites of applications (also known as “APIOps”). As platform engineering teams build observability into software development processes, customer aligned teams use observability to address pain points before they become complaints, including detecting changes to APIs.

Drivers

- API observability is key for API-first technical architectures.
- API observability can range from simple log analysis to sophisticated dashboards, from monitoring and alerting to automated analysis of important events, such as an API version change.
- As APIs proliferate, most are fronting complex microservices architectures or serverless functions. Understanding the dependencies among these systems is critical to all phases of system design, from implementation to incident management.
- OTel standards continue to evolve to aggregate views of APIs across traces, metrics and logs. OTel is an open-source framework for building observability into APIs. Many monitoring and observability vendors consume APIs that are built on OTel standards.
- Continued migration to cloud-based infrastructure is driving the need to track and manage APIs and the services that implement them, regardless of their infrastructures; however, many services rely on infrastructure events and logs for monitoring and alerting. Aggregated observability includes combining metrics from on-premises, hybrid and cloud-hosted APIs into a single, observable platform.
- APIs across large enterprises can be variably documented, developed and deployed creating challenges in discoverability and usability. Software engineers must untangle complex ecosystems with emergent behaviors. A single breaking change to an API can have effects across the ecosystem. API observability enables developers to understand ecosystem behaviors at a higher level of abstraction, supporting quicker value delivery.
- The need for higher quality and resiliency is driving the adoption of API observability practices. API observability enables engineers to understand how their APIs and services are performing locally, and how they will perform as part of the ecosystem. This allows bugs, performance issues and user experience (UX) problems to be caught early in the development life cycle.

Obstacles

- Although the techniques and practices of API observability can range from simple monitoring to sophisticated event analysis, describing the importance of observability can be challenging. Securing funding for API observability can be difficult, because a great deal of the data may be trivial and only occasionally operationally critical.
- The value of API observability improves as its scope increases from API design and development to testing and monitoring in production. Developing a platform to meet these needs can be complex and time-consuming.
- Standards that support API observability, such as OTel, are still developing, and it can be easy to make technology and standards adoption decisions that lock an organization into a particular vendor or technology ecosystem.

User Recommendations

- Explore API observability solutions that align with their overall observability strategies. Once a pattern is in place — such as monitoring as part of incident management — expand other observability practices to adjacent organizations or divisions.
- Make API observability the responsibility of a team with an adequate number of engineers working on the implementation of API observability platforms. Make the practice of API observability easy to adopt by value-stream-aligned teams by embedding it in a platform, run by a platform engineering team.
- The cost of observability platforms and approaches, whether “home-brewed” or purchased as a service, can vary greatly. Observability collects a large amount of information, which must be transported, stored and analyzed quickly under mission-critical pressure. Weigh the costs associated with these benefits carefully.

Sample Vendors

Chronosphere; Dynatrace; Moesif; New Relic Datadog; Optic; Postman (Akita Software); Tyk

Gartner Recommended Reading

Cool Vendors in Software Engineering: Improving Digital Resilience

Magic Quadrant for Application Performance Monitoring and Observability

Top Strategic Technology Trends for 2023: Applied Observability

Consider These Key Functional Areas for Application Performance Monitoring and Observability

How to Start and Scale Your Platform Engineering Team

Open APIs in Insurance

Analysis By: Sham Gill

Benefit Rating: High

Market Penetration: Less than 1% of target audience

Maturity: Embryonic

Definition:

Open APIs (aka public or external APIs) are application programming interfaces (APIs) that are published for consumption by third-party users and applications. Insurance open APIs enable consumers (developers) to use self-service portals to register and gain access to insurance products and services.

Why This Is Important

Open initiatives are paving the way for new innovation and possibilities for disruption in insurance, and APIs provide the technology that makes them work. It's important that insurance CIOs recognize that open APIs aren't just a technology issue. They enable connectivity between the insurance company and its external ecosystem partners. As such, they need to be considered as reusable products in their own right, with their own intrinsic business value.

Business Impact

Publication of their own open APIs and using partners' open APIs can boost digital transformation in insurance. Open APIs can expand an insurer's reach to a broader audience and enable innovation and transformation by providing frictionless value exchange with business ecosystem partners. They also enable faster delivery of new products, services and business models that enable the direct and indirect monetization of APIs.

Drivers

- The 2022 Gartner Financial Services Technology Survey reveals that the top three areas P&C insurance CIOs expect open APIs to make an impact are increasing innovation, introducing new products and services, and improving the customer experience (CX). Furthermore, 67% have either already invested and deployed, or are already experimenting with open APIs.
- Open APIs lay the foundations for open insurance, which is a natural evolution from open banking and the whole open finance movement. Open insurance will be driven by a market need for insurers to transform and participate in business ecosystems. This will require insurance CIOs to prioritize open APIs, whether inbound or outbound, in their digital technology platform roadmaps.
- They enable the creation of net new revenue streams through new business models, products and services distributed through partners. Embedded insurance is one such example where open/external APIs play a critical role in providing distribution partners with ease of access to insurance products.
- Changing consumer expectations on digital access to insurance require open APIs to meet the need to access and control personal data
- Open APIs support process optimization by reducing transaction friction between partners and customers. For example, enabling claims data, images and analytics captured at the first notice of loss to be seamlessly exchanged with auto repair partners to optimize estimation and repair scheduling.

Obstacles

- The strategic and business value of open APIs in insurance is not understood. Integrations are still being built using traditional approaches, like custom-built point-to-point product extensions, with associated traditional KPIs like the number of API calls.
- When compared to other financial services business divisions like banking, standard generation and acceptance in insurance has been far lower which obstructs scaling.
- While frictionless value exchange is technically possible, it may not be desirable for business or compliance reasons. For example, enabling insurance provider switching via open APIs may be achievable. But, switching can often be a difficult process that necessitates advice on issues like product fit and regulatory compliance.
- Open initiatives are growing across industries. Insurers probably use APIs more than they produce, thus they need to know which APIs they can use.
- Current use cases focus on specific parts of the insurance value chain, like data aggregation, quotes and new business, and predominantly on simpler P&C products like travel and gadget insurance.

User Recommendations

- Create the case for investment in open APIs by working with business stakeholders to identify the integration points in future open insurance business models that involve external partners.
- Track API transactions by using API management tools to monitor message volumes and trends, and provide business stakeholders with clear evidence of the need for open APIs.
- Foster relationships with external partners by identifying common transactions to help fund the co-creation of new open APIs.
- Entice partners to experiment with creating new open API-based services by creating sandbox environments that require less rigorous scrutiny and registration than production versions to reduce the barriers to adoption.
- Manage the consumption of third-party APIs the way that you should manage SaaS applications: negotiate and enforce SLAs, security compliance, licensing terms and continuity of service.

Gartner Recommended Reading

To Prepare for Open Insurance Opportunities, CIOs Should Strategically Invest in APIs

Case Study: An Insurance API-Driven Digital Ecosystem Transformation

How to Design Great APIs

How to Use KPIs to Measure the Business Value of APIs

Event-Driven APIs

Analysis By: Max van den Berk

Benefit Rating: Moderate

Market Penetration: 1% to 5% of target audience

Maturity: Emerging

Definition:

Event-driven APIs describe the interfaces used in event-driven architecture, including subscriptions and channel creation. Event-driven APIs differ from traditional request/response REST APIs because they enable asynchronous communication. Popular standards for event-driven APIs include AsyncAPI.

Why This Is Important

Event-driven APIs differ from traditional request/response APIs because they enable real-time notification of changes to data and application state. They are also provided as part of an event broker infrastructure.

Industries like financial services require timely updates of data and application state, which have driven the usage of event-driven architecture and event-driven APIs.

Integration scenarios also drive the usage of event-driven APIs, including data synchronization across SaaS services.

Business Impact

Event-driven APIs open up business opportunities for faster response to change and streaming analytics.

They also facilitate, and help standardize, event-driven architecture (EDA). Event-driven APIs bring advantages such as enabling push notifications, which are much more efficient and less expensive (from both a time and networking perspective) than polling.

Drivers

- Mobile applications involve the widespread use of events, such as webhook notifications using server-sent events (SSEs) delivered over HTTP.
- Data analytics and artificial intelligence drive new use cases related to events and notifications, which in turn drive the need for event-based APIs.
- Open-source and cloud event broker technologies, especially Apache Kafka, have raised awareness of publish-subscribe infrastructure, as well as accessibility to it, and have encouraged adoption of EDA.
- The OpenAPI Specification (OAS) version 3, introduced in 2017 and now reaching widespread adoption as a standard for publishing APIs, includes a provision for callbacks as a means to describe event-driven APIs. OAS version 3.1 (February 2021) adds support for webhooks, the most common event-driven API approach for internet-facing APIs. It should be noted, however, that webhooks are a one-to-one pattern, as opposed to EDA, which also supports a many-to-many pattern.
- AsyncAPI, which defines a standard for how event-driven APIs are published, is supported by the Linux Foundation, which is the same standards body that houses OAS.

Obstacles

- API mediation products, such as API gateways, are often built or configured only for request-response APIs. In some cases, the patterns of event-driven APIs (including fire-and-forget or publish-and-subscribe methods) are not supported in the API mediation product.
- Protocol support for event-driven APIs is diverse, including webhook (HTTP/S), WebSockets, MQTT, advanced message queuing protocol (AMQP) and Apache Kafka. This makes decision making more difficult for software engineering leaders.
- Although API portals are widely used for publishing request/response APIs, their usage for event-driven APIs is nascent.
- Reuse of schema definitions, such as Protobuf, Avro and graphQL, is not standardized, and might require tooling from the end user to support it.
- Event-driven APIs, like all event-driven architecture, introduce challenges for debugging and testing.

User Recommendations

- Evaluate whether your chosen API mediation products, including API gateways as part of API management, support event-driven APIs as part of your technology selection process.
- Adopt a community-of-practice approach to raise your organization's overall skills on event-driven APIs, and their relationship to event-driven architecture and stream processing and analytics. This may be a specific EDA community of practice, or part of an API community of practice.
- Identify opportunities to replace request/response APIs with event-driven APIs within your API portfolio to enable both internal and external integration scenarios.

Sample Vendors

Ably; Axual; Axway; Confluent; Gravitee; Postman; SmartBear Software; Software AG; Solace; TIBCO Software

Gartner Recommended Reading

Using Event-Driven Integration With Enterprise Applications

Maturity Model for Event-Driven Architecture

At the Peak

Financial APIs

Analysis By: Don Free, Mark O'Neill

Benefit Rating: Moderate

Market Penetration: 5% to 20% of target audience

Maturity: Adolescent

Definition:

Banking API aggregators provide financial data APIs that connect to multiple banks, thus simplifying banking integration for application developers. Financial data APIs provided by banking API aggregators typically include balance verification and funds availability APIs. Customers of banking API aggregators include fintechs, lenders and banks themselves.

Why This Is Important

Fintechs, lenders and app providers increasingly require access to bank accounts. Although open banking regulations exist in many countries that require banks to provide APIs, APIs remain unstandardized and developers often prefer to use an intermediary for simplicity. Banking API aggregators provide this capability, using banking APIs or in a customer-permissioned "screen scraping" approach. Increasingly, banking API aggregators provide emergent capabilities such as risk analysis and payment support.

Business Impact

Financial data APIs boost application development by simplifying access to bank data and services, but impact and usage differ among banking industry participants:

- Large banks — Prefer to directly control API access and not delegate access to banking API aggregators
- Midtier or small banks — Can avoid expensive API platforms by using data aggregators to publish financial data APIs
- Fintechs — Can leverage banking API aggregators to reduce the cost of integrating with each bank individually

Drivers

- A wide variety of organizations, including lenders, e-commerce sites and fintechs providing services, require access to banking data, which drives the need for banking API aggregators to provide this data. Open banking, banking as a service and embedded finance are often the source for these needs.
- APIs are now the preferred mechanism to integrate with banks due to widespread developer skills and tooling support. Banks also favor APIs due to the ability to apply security and traffic throttling.
- Many banks do not provide financial data APIs but where they do, they typically differ from each other. This drives the need for banking API aggregators to provide single APIs in front of multiple banks that may or may not have financial data APIs of their own.

Obstacles

- Some larger banks have reacted to banking API aggregators by blocking their connections. This is because they prefer partners to use the bank's own APIs directly.
- Privacy is a concern when banking customers share their online banking credentials with banking API aggregators as a part of account linking. In jurisdictions with open banking regulations, this concern is addressed by permission-based account linking without password sharing, rather than the use of screen scraping.
- Banking API aggregators typically operate in just one country or one region; however, this is beginning to change.

User Recommendations

- Identify and inventory the differing financial data APIs required to support various initiatives, such as open banking, banking as a service and embedded finance.
- Evaluate banking API providers if your organization requires access to banking services, such as account verification, across different banks.
- Question banking API aggregators on their security and privacy controls.
- Compare the pricing models of banking API aggregators before choosing a provider, since they vary on their pricing models.
- Establish a Chief of Information Flows or similar position to monitor and rationalize use of API provider offerings since many are expanding their transaction services.

Sample Vendors

Brankas; Envestnet | Yodlee; Fabrick; Finicity; MX; Plaid; Salt Edge; Tink; Token.io; TrueLayer

Gartner Recommended Reading

How to Evaluate API Management Solutions

API-Centric SaaS

Analysis By: Yefim Natis, Anne Thomas, Mark O'Neill

Benefit Rating: High

Market Penetration: 1% to 5% of target audience

Maturity: Emerging

Definition:

API-centric SaaS is a cloud application service designed with programmatic request/reply, or event-based, interfaces (APIs) as the primary method of access (instead of the traditional, and now optional, user interfaces). The strategic intent for API-centric SaaS is to contribute a set of business software components, packaged for use by advanced or business technologists as building blocks for composing custom application processes and services for end users.

Why This Is Important

API-centric SaaS serves as a foundation of creative innovation by business and software companies. It exposes modular business software as reusable building blocks in custom application development. Organizations create composite application processes and experiences that are more varied than relying entirely on in-house resources, and better targeted than relying entirely on the SaaS provider. Greater creativity in application engineering translates to business empowerment for faster, safer and more efficient innovation.

Business Impact

Business organizations equipped to use API-centric SaaS create new application experiences for their employees and customers, through composition of the new and prebuilt business software components, some sourced from multiple applications. They gain access to more impactful innovation to be more adaptive to the changing needs of the users and to better respond to competitive opportunities. Procurement of application services gradually becomes better matched to the consumed value.

Drivers

- Modern application design relies on cross-application integration and composition, compelling application vendors to deliver their business functionality, optionally or primarily, equipped for programmatic access.
- The growing popularity of the “headless” SaaS architecture in digital commerce provides the acceleration of creative innovation when using the modular API-first application design, changing the users’ assessment criteria for SaaS to favor the support of composability.
- The technology and skills for integration, including management of APIs, are widespread, promoting increasingly advanced use of programmatic interfaces to business applications.
- The demands for advanced customization of application experience in business organizations have evolved to the point that SaaS vendors must allow rearrangement of their business functionality by their customers. API-centric SaaS serves that purpose.
- Many older applications are increasingly accessed via APIs to include them in the modernization and innovation of organizations’ IT. This prepares organizations’ skills and technologies to include API-centric SaaS capabilities into their software engineering practices.
- Business application design has become significantly partitioned into the back-end functionality with its APIs and the front-end multiexperience, each side using different tools and design expertise. Some business-oriented application vendors find it convenient to concentrate on the back-end data and business logic, and leave the finalized user experience to separate teams, including the customer’s own developers.

Obstacles

- API-centric SaaS is a relatively new phenomenon. Both SaaS vendors and business developers may lack the required skills and tools.
- The best practices for pricing and procurement of API-centric SaaS are not well-developed, delaying adoption or increasing its costs. The pricing of some occasional use of APIs is common (and expensive) and does not match the use practices of API-first application products.
- Using multisourced API-centric components for assembling new application processes and experiences requires some integration work that may not be supported in selected composition tools. This requires advanced software engineering skills and delays adoption of API-centric SaaS by mainstream organizations.
- Reduced or absent user interfaces packaged with an API-centric SaaS assume and require that the customer implement their own differentiated application and user experience. What is a welcome opportunity for innovation for some can be a burden to others, delaying adoption of API-centric SaaS.

User Recommendations

- Build the tools and skills of API management that recognize the added requirements to govern access to imported third-party APIs.
- Give preference to SaaS offerings that expose and price more of their business functionality as APIs and/or event streams.
- Plan for the increasing use of composition and integration of API-centric business software in the design and delivery of application services, processes and experiences.
- Ensure clean API-based separation of the back-end business logic and the front-end user experience in most enterprise applications, to maximize the long-term benefits of adopted API-centric SaaS.
- Give preference to application platform offerings that are well-equipped for managed access to external APIs and event sources.
- Practice use and governance of APIs and event streams in preparation for greater adoption of API-centric SaaS.
- Watch for opportunities to experiment with a new business model by offering some of your business functionality packaged as priced API products or services.

Sample Vendors

Algolia; Alloy; Clearbit; Cloudinary; Lob; MessageBird; Plaid; Strapi; Stripe; Twilio

Gartner Recommended Reading

Accelerate Digital Transformation With an API-Centric Architecture for Enterprise Applications

How to Successfully Implement API-First Integration

Partner With Product Managers to Ensure the Success of API-Based Products

Banking Product Leader Insight: Think Beyond APIs to Address Composability

Quick Answer: What GMs Need to Know About the Composable Future of Applications

FHIR APIs

Analysis By: Mandi Bishop

Benefit Rating: High

Market Penetration: 5% to 20% of target audience

Maturity: Emerging

Definition:

HL7 Fast Healthcare Interoperability Resources (FHIR) APIs represent a modern standard for health data exchange between ecosystem participants such as payers, providers, life science companies, regulatory agencies, social services providers and patients. FHIR enables data sharing for domains including administrative, clinical and social determinants. FHIR is open source and based on widely adopted internet standards such as REST and JSON.

Why This Is Important

Health data interoperability has been elusive for decades, with proprietary standards dominating core systems and reporting requirements. FHIR APIs provide an open common standard for secure data exchange between health ecosystem participants such as providers, payers and life sciences organizations (collectively known as HCLS) as well as society stakeholders. FHIR undergoes continuous development and refinement by the global HL7 community — so it is extensible and adaptable.

Business Impact

FHIR APIs:

- Provide a common global standard for data exchange across HCLS industry sectors and other ecosystem participants that begins to meet the industry's interoperability goals.
- Accelerate solution development and implementation through widely adopted web standards such REST, XML, JSON, HTTP and OAuth.
- Establish standards-based information usage (such as clinical practice implementation guidelines) that improves collaboration and workflow integration between parties and regions.
- Decrease time to value for data-sharing initiatives.

Drivers

- Government interest in and support for health data standards are increasing in several global regions. Federal and state mandates in the U.S. require FHIR APIs for use cases such as giving patients access to their own data. The National Health System (NHS) in the U.K. uses FHIR for all new healthcare data interchange APIs. The X-eHealth Project is developing an FHIR implementation guide for cross-border information exchange across the EU. Brazil's Ministry of Health uses FHIR as the standard for its National Health Data Network. In 2023, Israel proposed legislation to mandate FHIR APIs and standard terminologies.
- The COVID-19 pandemic underscored the need for common open standards for health data exchange across HCLS sector and geographic boundaries. The easier it is for practitioners and researchers to securely share information, the faster and more accurately the world can respond to emerging infectious disease threats.
- Value-based care arrangements that span HCLS sectors are gaining traction. These arrangements require expansive data sharing, such as pharmaceutical manufacturers participating in chronic condition management and bearing risk for clinical outcomes.
- Open standards-based APIs will eventually reduce the complexity and cost of data exchange for healthcare administration by limiting laborious data translation needed between the proprietary standards of each entity to authorize services, review clinical documentation or pay claims.
- A thriving FHIR server solution market exists that includes open-source applications, every major cloud service provider and niche solution vendors. In addition to the FHIR servers, many vendors are FHIR-enabling solutions such as electronic health records (EHRs), claims engines, care management systems and population health analytics platforms.

Obstacles

- HCLS core clinical and administrative systems innovation happens slowly or not at all. Although FHIR is an open standard and free to use, substantial market or regulatory pressure is needed for vendors (particularly mature vendors with large market share) to support FHIR APIs before they achieve widespread adoption.
- HCLS organizations typically have sprawling data environments that would require heavy financial and resource investment if they are to adopt a new way of encoding health data.
- Technical challenges currently limit FHIR adoption at national scale. Intermediaries can interrupt rather than facilitate data exchange. Patient identity matching across stakeholders is unreliable. Authentication and authorization approaches vary, threatening privacy. No centrally maintained FHIR endpoint directory exists.
- Health data's value makes it a rich target for hackers. API security varies from system to system. EHRs are typically secure environments, whereas mobile apps may store API keys and tokens in clear text.
- FHIR standards are constantly evolving, creating a moving target for implementation and maintenance.

Analyst Notes: Due to high interest, a rapidly maturing vendor market and regulatory mandates pushing FHIR API adoption, we are introducing this innovation near the Peak of Inflated Expectations. However, substantial obstacles to achieving ROI exist — so it will take up to 10 years to achieve mainstream adoption globally. Adoption will be much faster in regions with strong regulatory requirements. Representative vendors have a global presence and offer FHIR services, as FHIR itself is open source.

User Recommendations

- Participate in FHIR workgroups that are defining use case-specific logical models and profiles as well as tackling technical challenges for scaling. These include the Da Vinci Project (for administrative use cases); the Gravity Project (for social determinants of health); Gender Harmony Project (for sexual orientation and gender identity); Project Vulcan (for clinical and translational research); X-eHealth Project (for EU cross-border interoperability); Helios (for public health); and the FHIR at Scale Taskforce (FAST — for resolving technical challenges).
- Evaluate existing core system and data platform vendors for their FHIR API support capabilities and roadmap plans.
- Incorporate FHIR APIs into your API management strategy.
- Collaborate with regional HCLS partners to identify high-value FHIR API use-case opportunities. Pilot (or expand, if existing) connections to establish baseline costs, your implementation time frame and initial performance goals.

Sample Vendors

Alibaba; Amazon Web Services; Google; IBM; InterSystems; Microsoft; Salesforce (MuleSoft); Smile Digital Health

Gartner Recommended Reading

Quick Answer: What Kind of Governance Does Healthcare Data Interoperability Need?

U.S. Healthcare Payer Interoperability Benchmarks, 2Q23

Establish Interoperable Application Ecosystems Early in Your Composable Healthcare Provider Roadmap

Event-Driven Architecture

Analysis By: Yefim Natis, Gary Olliffe, Max van den Berk

Benefit Rating: High

Market Penetration: 5% to 20% of target audience

Maturity: Adolescent

Definition:

Event-driven architecture (EDA) is a style of system design where components communicate indirectly by passing event notification messages via an intermediary (an event broker). EDA is a long-standing architecture model. The increased demands of digital business, including cloud-native architectures, globally distributed computing, stream analytics and real-time decision making, have reintroduced EDA as newly relevant to IT leaders and technology providers and placed it back onto the Hype Cycle.

Why This Is Important

EDA provides advanced opportunities for scale, extensibility and resilience in applications through its asynchronous, intermediated, pub/subdesign model. Monitoring business and technical events in real time enables continuous analysis of the context for advanced intelligence in decision management.

Business Impact

An event-aware organization is more responsive in its ecosystem — more empathetic in its customer experience and more intelligent in its decision making — than a purely transaction-centric business. Competence in EDA accelerates the transition to the modern, continuously intelligent and innovating adaptive-scale business. Lacking event awareness, organizations may struggle to support business requirements at competitive speeds, agility, context awareness and cost-efficiency.

Drivers

- Digital business demands real-time context awareness through stream analytics to support intelligent business decisions. Applications that adopt EDA become sources of such context and empower their business decision makers.
- Application designers that seek high agility and scalability turn to EDA to implement more flexible, extensible and scalable applications and data communications.
- Growing interest in composable architecture brings organizations to EDA to maximize the autonomy and efficient use of its modular components (packaged business capabilities).
- The popularity of Apache Kafka is creating greater awareness of EDA among mainstream organizations and their software engineering leaders.
- Many major application vendors, including Salesforce and SAP, upgraded support of EDA to their applications and application platforms in recent years, enabling more intelligent and introspective monitoring of business processes.
- All cloud hyperscalers have added or upgraded their support for EDA by adding and extending their messaging and event brokering services.
- Application integration continues to gain adoption in mainstream organizations, and EDA is a popular model for strategic integration design.

Obstacles

- The lack of productivity and governance tools dedicated to EDA limits the design of EDA-based applications to more advanced engineering teams, and thus delays broader adoption.
- The diversity of protocol, message and API formats and standards for event processing limits adoption and increases implementation costs.
- The design principles of EDA are less well-understood by most development teams because of the complex trade-offs associated with asynchronous communication and the familiarity bias in favor of the common and ubiquitous request/reply model, often implemented using REST APIs.
- Event-driven communications can deliver only eventual consistency. Applications that require synchronization of distributed database updates must choose a different architecture.

User Recommendations

- Develop an inventory of the currently deployed EDA-related technologies (such as queuing, message and event brokers) and practices; the existing technologies are likely sufficient for some or most of early EDA initiatives.
- Build a strategic roadmap for full adoption of EDA into your standard set of skills and capabilities.
- Adopt EDA gradually, as the industry develops required standards, best practices and improved product design and management tools.
- Aim to establish EDA, along with request-driven service-oriented architecture (SOA), as the common and complementary architecture patterns, both considered for future application initiatives.
- Combine technologies and practices of EDA with event streaming and analytics to amplify the business value of the architecture and justify the essential initial investment.
- Work with business stakeholders to coordinate the discovery and analysis of business events; aim for synergy in business and technical modeling of event-driven solutions.

Sample Vendors

Amazon Web Services (AWS); Confluent; Google; IBM; Microsoft; Oracle; Salesforce; Solace

Gartner Recommended Reading

Using Event-Driven Integration With Enterprise Applications

Essential Patterns for Event-Driven and Streaming Architectures

Maturity Model for Event-Driven Architecture

Innovation Insight for Event Thinking

Sliding into the Trough

CSP Open APIs

Analysis By: Amresh Nandan, Juha Korhonen, Ajit Patankar

Benefit Rating: Transformational

Market Penetration: More than 50% of target audience

Maturity: Early mainstream

Definition:

Open APIs refer to application programming interfaces that are publicly available, so developers can use them for programmatic access to proprietary software. Such APIs, often developed by a commercial entity, industry body or industry standardization/specification organization, are designed for ease of integration, utilization of data and functionalities, and monetization of assets/capabilities.

Why This Is Important

Communications service providers (CSPs) are increasingly adopting open APIs for better modularity, faster integration and reduced vendor lock-in. Integration and monetization requirements have forced detailed specification and standardization by industry bodies, such as the 3rd Generation Partnership Project (3GPP), European Telecommunications Standards Institute (ETSI), TM Forum, Metro Ethernet Forum (MEF) and GSM Association (GSMA).

Business Impact

With increase in modularity of applications, there is value in adopting open APIs for faster, easier and cost-effective integration. Open APIs can have a big business impact, but their success depends on their adoption level, which varies significantly across CSPs and vendors. Vendors' implementation of open APIs has increased due to CSPs' compliance requirements in their RFPs, leading to improved and faster integration of typical business support system (BSS) and operations support system (OSS) applications.

Drivers

- Open APIs in customer, product, service and resource management functions are leading than other functions, as CSPs have prioritized use of open APIs in their BSS and OSS application integration.

- A key reason for end-user adoption of open APIs is the desire to achieve greater freedom in sourcing and integrating technologies.
- In addition, such APIs by industry bodies are often specified or developed in a collaborative manner, leading to early identification of nuanced requirements and integration challenges.
- CSPs sourcing technologies from multiple vendors is the most appropriate way for faster and cost-effective interoperability.
- As CSPs look toward developing and participating in ecosystems, open APIs (such as GSMA Open Gateway) are a necessity to expose data and functionalities, while enabling simultaneous secure integrations with industry verticals and other partners.
- CSPs are increasingly tracking the level of TM Forum open API adoption by vendors through their RFIs and RFPs.
- The 5G network data analytics function (NWDAF) and network exposure function (NEF) have further boosted the idea of standard/open APIs, for the purpose of better management and monetization of the network.
- Some leading CSPs have also started focusing on open APIs for the purpose of monetization of data, intelligence and network assets through their service management/digital enablement platforms. We see this focus to intensify in the coming years.

Obstacles

- Despite continued efforts by TM Forum and other forums in developing a library of open APIs, there are sophisticated functionalities in some vendor solutions, for which no such standard and open APIs are available.
- CSPs have the intent to use standard and open APIs. Nonetheless, many CSPs lack a proper integration and API strategy, which leads to lackadaisical adoption.
- Continuation of old solutions and architecture in many CSPs is an obstacle for open APIs implementation, though some CSPs have been working on overlay mechanisms.
- Even though vendors' adoption of open APIs has increased, they continue to exhibit reluctance in implementing open APIs, unless forced by their key customers.

- Some vendors have developed their own set of APIs and project them as analogous to open APIs. Such an approach adds complexities and fragmentation to open APIs adoption.

User Recommendations

- Develop your integration strategy as a standard guideline for using standard and open APIs, and alternate mechanisms where such APIs are unavailable.
- Use adoption of open APIs by the vendors as a critical evaluation criteria during the vendor evaluation process.
- Negotiate the implementation of open APIs by the vendors in their implementation, as and when they are available, through contract terms and conditions.
- Dedicate resources for participation in API development initiatives, trials and pilot programs, as the effectiveness of such APIs take time and investments.
- Ensure access to development resources and create the developer support to have efficient use of open APIs to enable partners. Examples of such initiatives include TM Forum's Open API Catalyst programs, where many CSPs and vendors participate to address specific challenges.
- Participate in industry open API programs that offer CSPs ways to monetize network assets with ecosystem- or platform-based business models.

Gartner Recommended Reading

Objectives and Principles for OSS Architecture Evolution in CSPs

Market Guide for CSP Service Design and Orchestration Solutions

Market Guide for CSP Customer Management and Experience Solutions

Market Guide for CSP Revenue Management and Monetization Solutions

API Security Testing

Analysis By: Dale Gardner

Benefit Rating: High

Market Penetration: 20% to 50% of target audience

Maturity: Adolescent

Definition:

API security testing is a specialized type of application security testing (AST) aimed at identifying vulnerabilities in APIs. Checks should include both traditional application vulnerabilities (such as injection attacks) and API-specific issues (such as broken-object-level authorization). Availability of an API specification (such as OpenAPI) is sometimes a prerequisite for effective testing. Discovery technologies help ensure that unknown APIs are identified and tested for vulnerabilities.

Why This Is Important

APIs represent a major attack surface for web-enabled applications. Attacks on and abuse of APIs result in serious adverse consequences, including data breaches and other security incidents. DevSecOps teams focus on the need for API security testing in development to prevent this, but APIs pose unique risks. Many organizations rely on traditional AST tools extended to support these requirements or speciality tools designed for APIs. Some vulnerabilities may require manual testing to detect.

Business Impact

APIs are a foundational element of many organizations' digital transformation efforts. Hence, securing APIs from attack and misuse is a continuing concern for many security and risk management (SRM) professionals. API-specific testing — pre- and postdeployment — builds a solid foundation for an overall API security strategy. Automated API discovery is needed because many organizations struggle to maintain an inventory of APIs and need help locating them so they are tested and managed.

Drivers

- In support of digital transformation efforts, APIs have become common in application architectures to enable information flow and support transactions between processes, applications and systems. However, that growth has brought increased attention from attackers, and APIs have become the primary attack surface for many systems.
- API attacks have resulted in a stream of data breaches and other security incidents, yielding significant damage to organizations and individuals. As a consequence, DevSecOps teams — along with the business leaders whose applications are supported by APIs — are increasingly interested in API testing and security.
- Because the creation, development and deployment of APIs may be loosely managed, security teams often have to contend with undocumented or shadow APIs, which exist outside normal processes and controls. Such APIs may be especially deficient in secure design and testing, posing an even greater risk. Hence, the discovery of APIs deployed to production is another important need.
- API security testing helps avert the tangible and intangible costs associated with breaches and other types of security incidents.
- Traditional AST tools — SAST, DAST and interactive AST (IAST) — were not originally designed to test for some of the unique types of vulnerabilities associated with typical attacks against APIs, or for newer types of APIs (such as GraphQL). API-specific vulnerabilities and modern API formats prompt security and development teams to implement specialized API security tools that are focused on testing, discovery of shadow APIs and protection from threats (or some combination of the three).

Obstacles

- Traditional tools offer inconsistent support for detecting API-specific vulnerabilities. APIs are susceptible to most traditional application attacks, but other attacks are common. For example, improper authorization checks around object identifiers have been cited in a number of breaches. Specialized tools or penetration testing may be needed for reliable detection of these flaws.
- Testing tools may require an API specification to perform effective testing.
- Testing tools may not support all API protocols. SOAP remains in widespread use, although it is being supplanted by REST APIs. GraphQL-based APIs are increasingly common, so additional support is required from tool vendors for effective testing.
- Some confusion remains in the marketplace, with various types of companies offering products (including traditional AST vendors and API testing and protection vendors), complicating evaluation and selection efforts.

User Recommendations

- Begin evaluation and selection efforts by focusing on the criticality of APIs; their security and business risks; and the technical requirements they pose.
- Examine the testing and discovery capabilities provided by existing tools in your application security portfolio. Many have added support for APIs, although actual tests may still focus primarily on traditional application vulnerabilities and lack support for API-specific vulnerabilities.
- Evaluate newer alternatives where gaps are found. They may also offer additional options, including audit of design-stage API specifications, discovery, vulnerability scans and threat protection.
- Complement automated testing tools with penetration testing services for comprehensive coverage as traditional AST is often unable to detect some common API-specific vulnerabilities.
- Evaluate the ability of a given tool to address multiple requirements. API security tool capabilities — including discovery, testing and threat protection — often overlap.

Sample Vendors

42Crunch; APIsec; Contrast Security; HCLSoftware; Noname Security; OpenText; PortSwigger; StackHawk; Synopsys; Veracode

Gartner Recommended Reading

API Security: What You Need to Do to Protect Your APIs

API Security Maturity Model

How to Deploy and Perform Application Security Testing

Research Index: Everything You Should Do to Address API Security

Critical Capabilities for Application Security Testing

GraphQL APIs

Analysis By: Andrew Humphreys, Dave Micko

Benefit Rating: Moderate

Market Penetration: 5% to 20% of target audience

Maturity: Adolescent

Definition:

GraphQL enables API consumers to use a standard syntax to specify the data attributes they want, meaning they can select to receive only the information they need. This contrasts with a traditional REST API, where the structure of the data returned is predefined. Many GraphQL APIs, such as the Salesforce GraphQL API, provide an abstraction over multiple distinct data sources and APIs. This way, they simplify retrieving data from multiple sources.

Why This Is Important

GraphQL enables flexible self-service access to data for front-end applications. It optimizes data movement between remote clients and back-end services by avoiding over- and underfetching issues. GraphQL also enables organizations to model, expose and use valuable data or metadata associated with important relationships between data entities. Organizations including Netflix and Airbnb are adopting GraphQL as a means of composing a large graph schema from independently managed subgraphs.

Business Impact

GraphQL enhances user experience development. It is a more efficient use of resources, compared to the overhead of individual REST API calls. Further, it enables consumers to explore and unlock data based on their individual needs, rather than the provider-defined REST APIs resource model. A GraphQL schema can also become a shared data model that is decoupled from the physical data and interface models of individual services. This can help establish a common understanding of business data.

Drivers

- Web developers — particularly ReactJS developers — are driving demand for GraphQL to enable self-service access to data and services. GraphQL allows API consumers to define the structure for the responses they require from APIs, enabling greater flexibility than REST APIs.
- GraphQL reduces the need for multiple API calls to access all the data a developer requires, and reduces the amount of unnecessary data returned in an API call.
- Vendors now provide GraphQL APIs that enable data access across multiple applications. API management products are increasingly adding GraphQL support, simplifying the adoption of GraphQL APIs.

Obstacles

- GraphQL APIs place significantly more burden on teams supporting APIs and require investment in new design, security, monitoring and governance skills. In addition, vendor support for GraphQL is still not as widespread as support for REST APIs. Security, governance and performance optimization practices are still maturing.
- Performance is a consideration due to the processing required to process a query and efficiently retrieve the data specified. This may require multiple back-end API requests or database queries.
- The availability of design and development skills for GraphQL APIs is currently behind the availability of skills for delivering REST APIs. For example, error handling in GraphQL is different and may add complexity the developer needs to plan for.

User Recommendations

- Evaluate GraphQL APIs when they are available from their platform providers, and adopt them to provide flexible self-service access to data, including for applications such as dashboards and mobile apps.
- Consider building GraphQL APIs to meet specific demand from (mostly front-end) developers. For example, demand could include situations where multiple consumers need different data payloads, and when consumers need to span multiple API's.
- Replace dedicated "experience APIs" or backends for frontends (BFFs) with GraphQL APIs that tailor API experiences to different API consumers. GraphQL APIs offer the consumer more flexibility and control over how related data is accessed than traditional APIs.
- Adopt a community of practice approach, as well as mentoring, "Golden Paths" and guidelines on internal developer portals. These initiatives will help spread GraphQL skills in your organization, as GraphQL APIs introduce new challenges and ways of thinking for both API consumers and API providers.

Sample Vendors

Apollo GraphQL; AWS (AppSync); Hasura; Hygraph; IBM (StepZen); RapidAPI; Tyk; WSO2

Gartner Recommended Reading

Choosing an API Format: REST Using OpenAPI Specification, GraphQL, gRPC or AsyncAPI

API Industry Standards

Analysis By: Andrew Humphreys

Benefit Rating: Moderate

Market Penetration: 5% to 20% of target audience

Maturity: Adolescent

Definition:

API industry standards describe best practices and conventions that should be followed, and technical standards that should be adhered to, when implementing APIs for common industry-specific use cases. They enable consistency in implementation, save design effort and improve interoperability. API standards are mature within some industries like banking and healthcare, are becoming mainstream in others like container shipping, and are generally emerging or adolescent in most other industries.

Why This Is Important

Implementing APIs that adhere to industry standards simplifies adoption, accelerates development and encourages innovation. Industry API standards provide the technical and, in some cases, the legal framework for APIs that enable the sharing of data and services. They are for improving integration between industry members, or for third parties to provide technology, a service or an app that makes use of the shared data and services, opening new business opportunities for API providers.

Business Impact

Industry API standards improve interoperability and help open new revenue streams and find new ways to communicate and engage with customers. For example, common industry API security standards in banking have opened up the financial services market to developers. Developers can use a bank's APIs to securely access a customer's information with their permission and build new offerings like managing multiple bank accounts from a single app and transferring money between them easily.

Drivers

- Government mandates are driving the use of API standards in healthcare (e.g., U.S. Office of the National Coordinator for Health Information Technology [ONC] regulations) and banking (e.g., the revised Payment Services Directive [PSD2] in the European Union, the U.K.'s Open Banking standards, and other open banking regulations worldwide).
- Industry standards provide templates that can be followed for implementing commonly required APIs, reducing design and implementation effort.
- API standards in industries like banking and healthcare have opened up access to data and services in a consistent way. This has enabled developers to deliver new applications that combine data from multiple API providers more quickly. For example, the Fast Healthcare Interoperability Resources (FHIR) standard in healthcare simplifies the sharing of a patient's healthcare data between healthcare providers and so improves the customer experience.
- Other industries, like container shipping with the Digital Container Shipping Association (DCSA), are looking to define industry standards for APIs to replicate the benefits from the improved customer experience, simplified interoperability and increased innovation seen in banking and healthcare.

Obstacles

- In many industries, there is a lack of successful initiatives to define API standards. Even in industries where certain standards are mandated, the definitions of actual published APIs may differ. This presents an obstacle to adoption, as well as to interoperability.
- There is not as strong a requirement for industry API standards in industries where there are not regulatory requirements or strong interoperability needs. The interoperability benefits require a critical mass of adoption. Early adopters may not see benefits initially.
- Vendor support for industry API standards is inadequate, though slowly growing. Using industry standards is not always free as you might need to join the organization that oversees them.
- Industrywide standards are often generic and tend to be more complex as they need to deal with the aggregate of all problems they might be applied to. This means they are not often the best examples of well-designed, consumer-friendly APIs and may not meet your specific needs.

User Recommendations

- Follow industry standards if you are in an industry like banking or healthcare where they are well established. This will enable you to benefit from the increased interoperability and innovation.
- Check the rate of adoption of API standards in your industry if they are not widespread to determine if use is common enough to provide additional value.
- Follow general, non-industry-specific API standards. These enable common understanding and interoperability between API providers and consumers. API standards include guidelines for how APIs are described and published. Standards such as OAS, OAuth 2.0 and OpenID Connect are relatively mature. Other API standards such as AsyncAPI, gRPC and GraphQL schemas are in a growth phase.
- Use aggregator API services, such as Plaid or TrueLayer in banking, which provide a single API in front of multiple back-end APIs, enabling adoption of standard APIs without requiring significant rework in back ends.

Gartner Recommended Reading

Choosing an API Format: REST Using OpenAPI Specification, GraphQL, gRPC or AsyncAPI

Market Guide for API Gateways

API Threat Protection

Analysis By: Mark O'Neill, Jeremy D'Hoinne

Benefit Rating: High

Market Penetration: 5% to 20% of target audience

Maturity: Adolescent

Definition:

API threat protection is the defense of web APIs from exploits, abuse, access violations and denial of service (DoS) attacks. API gateways, web application and API protection (WAAP), and specialist API security tools provide API threat protection through a combination of content inspection of API parameters and payloads, traffic management, and traffic analysis for anomaly detection.

Why This Is Important

Applications use APIs to access data and to call application functionality. APIs are easy to expose, but difficult to defend. This creates a large and growing attack surface, leading to a growing number of publicized API attacks and breaches. Successful attacks on APIs result in data breaches, leading to loss of sensitive data as well as reputational harm.

Business Impact

Because APIs are typically used for access to data or application functionality, often linked to systems of record, the impact of an API breach can be substantial. Privacy regulations typically require reporting if private data is breached through an unsecure API. APIs are easily and intentionally programmable, so a vulnerability can leak large volumes of data. The challenge of distinguishing malicious access from valid access further complicates the task of securing APIs.

Drivers

- Publicized API breaches are driving awareness of the need for API threat protection.
- Financial services APIs are high-value by nature, and vulnerable to abuse and fraud.
- APIs are widely used by large language models (LLMs) for data access, leading to an increased need to protect that access to data.
- Some established vendors have begun to use machine learning to detect potentially harmful API usage patterns and thus protect APIs from threats. These include WAAP vendors.

Obstacles

- Many organizations lack visibility of their APIs, as many are used as part of web or mobile applications and not published directly. This means that a key requirement of API threat protection is API discovery. The main obstacle to API discovery is that the APIs used by an organization are typically dispersed across multiple platforms, including cloud services.
- Organizational ownership of API threat protection is a challenge. Whereas the security team, under a CISO, typically manages a WAF, API gateways are managed by API teams. This can lead to API threat protection being neglected due to a lack of expertise.
- Many API security issues are related to business logic. Protection against business logic threats is difficult to automate completely because the protection tool needs to understand the logic and identify unusual usage.
- Behavioral anomaly detection is, by nature, a generator of false positives. Despite the growing use of generative AI, some API threat detection tools currently require human intervention to process false positives.

User Recommendations

- Discover and categorize your APIs before implementing threat protection in runtime.
- Implement a layered API security model. At the outer (typically cloud-based) layer, use volumetric distributed denial of service (DDoS) protection and bot detection. Behind this layer, apply API-specific authentication, authorization and traffic management.
- Assess the API protection provided by your current WAF or API gateway. If it provides immature or insufficient API protection, then investigate API threat protection specialist vendors.
- Ensure that the API protection rules in your chosen API threat protection solution are adaptable, based on the nature of the API itself. Static rate limits or IP allow/block lists are rarely useful in production environments or at scale.

Sample Vendors

42Crunch; Akamai (Neosec); Cequence Security; FireTail; Imperva; Noname Security; Salt; Threat Hunter (China); ThreatX; Traceable

Gartner Recommended Reading

Research Index: Everything You Should Do to Address API Security

API Security Maturity Model

Innovation Insight for API Protection

API-Based Digital Commerce

Analysis By: Aditya Vasudevan

Benefit Rating: Transformational

Market Penetration: 5% to 20% of target audience

Maturity: Adolescent

Definition:

API-based digital commerce (also commonly known in the market as “headless” digital commerce) is the use of APIs to decouple front ends from core commerce services and to integrate commerce capabilities within any touchpoint where selling is required.

Why This Is Important

The proliferation of touchpoints requires a multichannel, multiexperience approach to applications. This, in turn, requires the decoupling of the presentation layer from commerce services that an API-based approach offers. Some vendors provide API-based commerce platforms, while others retain a native storefront but also provide full APIs for headless operation, known as “head optional.”

Business Impact

Decoupling the front end is a step toward a modern, modular commerce architecture that provides business flexibility and IT agility. As commerce journeys become multiexperience, this is an enabler for delivering consistent experiences across all touchpoints. The storefronts must be delivered independently of the commerce application, via a digital experience platform (DXP) or custom front end. Although being API-enabled does provide ultimate flexibility, it requires digital maturity to succeed.

Drivers

API-based digital commerce adoption is driven by:

- Midsize to large organizations, that are looking to move up in the digitally mature scale. It has become the standard approach for the delivery of experiences by the enterprise, even if the underlying application remains monolithic.
- Recognition of the quality of digital experience as a key differentiator across multiple touchpoints (for example, native mobile apps, marketplaces, social platforms, in-store experiences, the Internet of Things, wearables, smart homes and vehicles).
- Maturity of front-end frameworks, single-page application (SPA) and progressive web application (PWA) as the dominant "next generation" of client-side, JavaScript-based presentation and the emergence of digital experience platforms (DXPs) requires digital commerce platforms to be fully API-enabled.
- Growth in DXPs supporting "experience-driven commerce."
- Commerce as an enabling part of a wider digital business technology platform.
- Pace of innovation in digital commerce driving more flexible, modular architectures.
- Expense and complexity of some leading "monolithic" commerce platforms, when a more agile, flexible subset of capability is desired.

Obstacles

API-based digital commerce improves business agility. However:

- The "headless" buzz has caused some disillusionment among those expecting it to be easy to achieve. Despite this, it is rapidly moving toward mainstream acceptance.
- Commerce experiences can be more complex to implement than single-vendor "full stack" solutions due to increased emphasis on integration efforts and governance of the decoupled front-end technology.
- A key challenge is ensuring business users retain no-code control over the storefront(s). This adds complexity to implementations and the business UIs required to support presentations and processes.
- Most vendors' native commerce platform storefronts are shifting from server-side "themes" or template engines toward client-side SPA/PWA. Thus, some of the benefits of API-based digital commerce are becoming available from almost all vendors. But having an API is not the same as being "API first," and not all the benefits of this new approach have yet to be realized.

User Recommendations

Pursue API-based commerce if you think it may fit your requirements, specifically if you:

- Want to retain granular control over multiexperiences, including by deploying a DXP, building via a multiexperience development platform (MXDP), or developing a SPA/PWA presentation tier.
- Are looking to support multiple digital and physical channels equally from the same business logic, and to support cross-channel continuity of experience.
- Already have (or are looking to implement) a DXP to provide a more consistent customer experience across commerce, brand and other digital properties.
- Have a large, inflexible, legacy, monolithic, full-stack commerce application that cannot be replaced in a single step, and want to migrate to a modular architecture.

Sample Vendors

BigCommerce; commercetools; Elastic Path Software; fabric; Infosys Equinox; Kibo Commerce; Shopify Plus; Spryker Systems; Virto Commerce; VTEX

Gartner Recommended Reading

Magic Quadrant for Digital Experience Platforms

Choose the Right Digital Commerce Platform Architecture

Quick Answer: What Does a Technology Reference Model for Digital Commerce Look Like?

Defining the Digital Experience Platform

API Testing Services

Analysis By: Jaideep Thyagarajan

Benefit Rating: High

Market Penetration: More than 50% of target audience

Maturity: Early mainstream

Definition:

API testing services are a set of outsourced testing activities that are performed directly and as a part of integration testing to validate whether the programming interfaces meet expectations for functionality, reliability, performance and security. Focusing mainly on the business logic layer of the software architecture, API testing uses software to send calls to the API, receive output and validate the system's response.

Why This Is Important

Defective APIs directly impact digital experiences, such as mobile apps and reactive web applications, meaning that API quality is critical. Poor-quality APIs impact developer experience, which is increasingly important as organizations use APIs internally and build external developer communities. Therefore, API testing is an important consideration to mitigate business risk and to ensure seamless integration of applications with other services.

Business Impact

APIs have emerged as a critical tool for building modern applications, as evidenced in the surge in organizationwide API strategies. An API that fails to deliver the expected level of quality, security, reliability and performance can thus have tremendous business impacts, both to the organization producing it and to those consuming it. Risks associated with application failure have broader business impacts; hence, the quality of the APIs produced and consumed is now more important than ever.

Drivers

- Many organizations have created API platform teams composed of API center of excellence (COE) team members with a mandate to deliver high-quality, reliable APIs.
- Organizations increasingly rely on APIs as a part of their daily operations, such as logistics APIs for retail deliveries, and APIs into systems of record. These APIs must be reliable and well-designed.
- APIs are treated more like products than code. They are designed for consumption for specific audiences (for example, mobile developers), documented, and versioned in a way that users can have certain expectations of their maintenance and life cycle. Because of better standardization, they have a much stronger discipline for security and governance, and can be monitored and managed for performance and scale, thereby amplifying the importance of testing them.
- API testing offers an opportunity to test the core functionality of the app without having to interact with a potentially disparate system. This helps in early bug detection, instead of bugs becoming larger issues during GUI testing, and thereby protects the application from malicious code and breakage.
- APIs have become a primary attack surface for many systems. These attacks have resulted in an endless stream of data breaches and other security incidents, yielding significant damage to organizations and individuals. As a consequence, software engineering teams — along with the business leaders whose applications APIs support — express significantly increased interest in API testing and security.
- Traditional testing skills and team culture do not apply naturally to APIs because APIs are used by other systems — not directly by end users. API testing calls for a different set of skills that involves the ability to understand the business logic of the service in addition to scripting and coding skills. Therefore, seeking specialized API testing services is becoming a matter of priority for many software engineering leaders.

Obstacles

- APIs present challenges like broader attack surface area, higher potential for unexpected misuse and unpredictable demand, among others. These challenges translate to specific testing ramifications that require strong consulting skills to factor in all of the API test scenarios, which some providers may struggle with.
- API testing includes tasks like preparing the environment on which the API will exist, updating the API testing requests scheme, deciding on the sequence of API calls and validating parameters, among others. These activities often call for a strong involvement of client-side resources as well.
- Incumbent vendors offering traditional testing services may not have strong API testing skills. So, the selection criteria need to factor in API-testing-specific key capabilities. These can include the ability to understand basic rules of creating good APIs, to design tests that are relevant for APIs, to use API testing tools, to understand business logic of service and to locate real user scenarios.

User Recommendations

- Begin evaluation and selection efforts by assessing the overall role that APIs play in your application portfolio, their criticality to the organization, technical requirements, and the security and business risks they pose.
- Prefer API testing services that are underpinned with intelligent test creation and automated validation. Because testing a broad range of conditions and corner cases is critical with APIs, automation must come to the forefront.
- Ramp up the scope for extensive performance testing as part of services. Considering the highly exposed nature of APIs, there is a strong potential for unpredictable and volatile traffic volumes. Therefore, it is critical to determine whether your APIs satisfy SLAs in the event of surging demand.
- Examine the testing capabilities provided by existing tools in your application testing portfolio, including full-life-cycle API management platforms, which may include API testing.

Sample Vendors

APImetrics; Applause; Cigniti; Infosys; Postman; SmartBear; Stoplight; Tata Consultancy Services (TCS); Tricentis; Wipro

Gartner Recommended Reading

How to Deliver Sustainable APIs

How to Successfully Implement API-First Integration

The Evolving Role of the API Product Manager in Digital Product Management

API Marketplaces

Analysis By: Andrew Humphreys

Benefit Rating: Moderate

Market Penetration: 5% to 20% of target audience

Maturity: Adolescent

Definition:

An API marketplace is a platform to share APIs. Consumers, mainly developers, use API marketplaces to discover APIs and, in some cases, may purchase access to them. They can be either public commercial marketplaces with APIs from multiple providers, public with APIs from a single provider, or private marketplaces for promoting an organization's internal APIs.

Why This Is Important

API marketplaces enable organizations to publicize their APIs. Marketplaces are usually associated with external marketplaces, which share APIs with a community of developers and enable partners to implement solutions using the APIs. However, as most APIs are meant for consumption by teams within an organization, marketplaces are more frequently internal. They make it easier to find APIs internally, helping with wider sharing of capabilities between different business units and product and development teams.

Business Impact

API marketplaces increase developer visibility and consumer mind share, drive API usage, and, by extension, increase business impact. API consumers can use marketplaces to simplify finding and comparing different APIs when they are looking for specific functionality but have not selected exactly which API to use. There is typically a cost involved with listing in a public API marketplace, but the benefits include exposure to a larger number of API consumers and access to features to enable monetisation.

Drivers

- The number of APIs within an organization is climbing, driving the need for developers to more easily discover which APIs and services are available.
- Composable business, including composable commerce, relies on the use of API marketplaces to share APIs and packaged business capabilities.
- Increased use of low-code platforms, integration platforms, robotic process automation (RPA) and analytics tooling enables more citizen development using APIs that may be sourced from API marketplaces.

Obstacles

- Public API marketplaces that provide a public directory of APIs from multiple providers have had disappointing results, as developers are more likely to go directly to API providers to sign up for APIs. This has resulted in API marketplaces in the Trough of Disillusionment. However, internal API marketplaces have had more success, since they enable developers to share APIs across multiple teams.
- API portals provided as part of API management platforms are typically basic in nature, resulting in significant customization work to create a customer-oriented API marketplace based on such an API portal.
- New open-source platforms, such as Backstage from Spotify, are driving the creation of internal API catalogs as part of larger developer hubs. If your developers are collaborating on solutions around their APIs already, then a simple catalog may be sufficient and a full marketplace is probably overkill.

User Recommendations

API providers:

- Create an internal API marketplace, focused on the needs of software engineers to share APIs across the organization.
- Examine billing terms to understand what the cost of using the marketplace is when considering commercial API marketplaces.
- When considering a commercial API marketplace, examine listing fees and value to your organization before committing.

API consumers:

- Ensure that you use APIs from trusted marketplaces and trusted API providers, examining usage agreements, licensing and billing terms carefully.
- Investigate whether consuming an API directly from the API provider offers better pricing or usage terms than consuming the API via a marketplace.

Sample Vendors

Achieve Internet; Bump; Postman; Pronovix; Readme; Smartbear (Swagger); Spotify (Backstage); Stoplight

Gartner Recommended Reading

Innovation Insight for Internal Developer Portals

Reference Model for API Management Solutions

Climbing the Slope

Microservices

Analysis By: Anne Thomas

Benefit Rating: High

Market Penetration: 20% to 50% of target audience

Maturity: Early mainstream

Definition:

A microservice is a tightly scoped, highly cohesive, strongly encapsulated, loosely coupled, and independently deployable and scalable software component. Microservices architecture (MSA) applies the principles of service-oriented architecture (SOA), DevOps and domain-driven design to the delivery of distributed applications. MSA has four core objectives: increasing agility, enabling autonomous teams, improving deployment flexibility and supporting precise scalability.

Why This Is Important

Microservices architecture enables autonomous teams and improves agility, flexibility, resilience, and precise scalability. It is a way to build cloud-native applications, and it facilitates continuous delivery practices. However, the architecture is complex, with disruptive cultural and technical impacts. Software engineering teams have been observed to use MSA indiscriminately, leading to overly complex architectures that fail to deliver anticipated benefits and often make things worse.

Business Impact

MSA increases business agility by enabling teams to isolate new feature deployments and regression impacts in their software products, in response to changing business requirements. It improves the scalability of the software engineering organization by enabling small teams to work autonomously to deliver different parts of an application. MSA allows teams to deploy a single feature without the delay, cost and risk of deploying the entire application.

Drivers

- **Continuous delivery:** Software engineering teams adopt MSA to facilitate a continuous delivery practice. The architecture must be combined with robust agile methods and strong DevOps practices to enable teams to safely develop and deploy small, independent features to production systems at the frequency at which they are delivered.
- **Autonomous teams:** When applied properly, MSA increases the independence of different parts of a large application. This enables multiple development teams to work autonomously and on their own schedules.
- **Cloud-native architecture:** MSA facilitates the building of cloud-native applications that support robust scalability and resiliency requirements.
- **Containers:** Microservices are frequently deployed in container management systems, which enable improved automation of delivery pipelines. Container management systems can also dynamically scale service instances in response to load requirements, and automatically recover services that have failed.
- **Resiliency:** When combined with resiliency patterns and practices, MSA enables the creation of self-healing systems that can continue to operate through partial outages.

Obstacles

- MSA and its benefits are often misunderstood, and teams may struggle to deliver outcomes that meet senior management expectations. Microservices should not be shared and they will not save you money.
- MSA delivers the most value when combined with continuous delivery practices. Management may be disappointed with the MSA cost-benefit equation, if teams are not doing continuous delivery.
- MSA is complex. Developers must acquire new skills, and adopt new design patterns and practices to achieve its benefits. Additional complexity also impedes incident recovery.
- MSA disrupts traditional data management models. Data consistency must be managed in service and interface design, rather than in data stores.
- MSA requires new infrastructure that increases the cognitive load on engineering teams.

- Many software engineering leaders underestimate the cultural prerequisites, such as mature agile and DevOps practices, and team structures aligned with service domains.

User Recommendations

- Set clear expectations by defining business goals for MSA adoption, based on realistic cost-benefit analysis.
- Improve outcomes by creating guidelines for where and when software engineering teams should and should not use MSA.
- Create platform teams to manage the development and runtime platforms that support microservices. This would reduce the cognitive load on developers and automate delivery of new microservices.
- Ensure that application architecture is as simple as possible to achieve your goals.
- Use MSA where appropriate to attain those goals. View microservices as a tool, not as a target architecture.
- Avoid MSA for small, simple, low-volume or static applications, where the cost of the increased complexity exceeds the benefits of improved agility, flexibility, or scalability.
- Address cultural concerns by aligning teams along business domain boundaries, investing in distributed computing architecture skills and improving DevOps practices.

Gartner Recommended Reading

Leading Teams to Success With Microservices Architecture

Solution Path for Applying Microservices Architecture Principles

10 Ways Your Microservices Adoption Will Fail — and How to Avoid Them

API Access Control

Analysis By: Nathan Harris, Erik Wahlstrom

Benefit Rating: High

Market Penetration: 20% to 50% of target audience

Maturity: Early mainstream

Definition:

API access control is a critical API security capability that provides authentication and authorization for APIs to deliver required security risk mitigation for access related threats. At a minimum, this involves an OAuth 2.0 authorization server, which supports and implements consent, manages authorization through scopes and associated claims, and issues customizable JSON Web Tokens (JWTs) to web servers, mobile apps, modern web apps and services used to access APIs.

Why This Is Important

Most web traffic now happens through APIs, rather than traditional web applications (i.e., web browser traffic), and the attack surface of APIs is now larger than the user interfaces (UIs). Along with this, attacks exploiting API vulnerabilities continue to increase. Organizations looking to mitigate the effects of this broadened attack surface should add authentication and authorization controls to API process flows when deploying API security.

Business Impact

API access control capabilities are important in several scenarios:

- Developer use cases, offering libraries, out-of-the-box integrations and best-practice examples of API access controls.
- Enabling user-present (including browser-based, single-page apps and mobile apps) and machine-to-machine access to API targets.
- Workforce and customer use cases, including customer data protection/privacy and consent management.
- Architecture standardization with central access controls for APIs.

Drivers

- API access control adoption has benefited from the popularity of access management tools (with market share adoption increases of more than 35% since 2020). Access management tools providing OAuth 2.0 have made the protocol an industry standard, driven by the continued increase in API delivered versus browser delivered functionality.
- Organizations' digital transformation efforts and need to provide secure API access for customer engagement journeys (CIAM) drive improved API access control strategies.
- The application attack surface has grown with the increased use of exposed APIs. Gartner estimates that more than 90% of web-enabled applications have more surface area for attack in the form of exposed APIs, rather than the user interface (UI).
- Lack of advanced IAM capabilities of API gateways (e.g., advanced adaptive access and support for evolving identity standards) has increased demand for specialized access control for APIs, delivered by AM, externalized authorization management and other IAM and API security specialist tools.
- Beyond OAuth 2.0 authorization servers, IT leaders also need the ability to centralize authentication and authorization services for libraries, sidecars and out-of-the-box integrations with APIs and API mediators.
- Developer self-service capabilities are also increasing in demand, for managing apps and services and support for flows such as OAuth mTLS, JWT and SAML grant types; the device flow; token exchange; introspection; and revocation. Authorization servers also provide strong, agile cryptographic mechanisms for signing tokens.
- Modern applications and service patterns (e.g., service mesh and REST) are adopting JWTs as authentication and authorization mechanisms. JWTs are becoming the de facto standard to convey authorization information and claims about users and services, enabling a scalable, privacy-preserving, zero trust architecture that helps developers quickly deploy protected services.

Obstacles

- Time to market for features/functions is often seen as more important than applying rigorous API protection.
- A lack of generally accepted API access control best practices, along with a lack of developer and security staff training in IAM best practices.
- The belief that API gateways (or any single tool) are sufficient to enable and securely run API-based applications.
- Lack of tooling that enables scaling centralized policy authorizing and continuous policy life cycle management to an increasingly diverse set of API targets and mediators that enforce these policies.

User Recommendations

- Define a comprehensive API access control strategy by broadening tool selection. This requires authorization servers, externalized authorization managers, secrets managers, API mediators and other in-line proxies. This is delivered by a combination of access management, API gateways and other specialized API security and IAM tools. In other words, plan for an identity fabric architecture approach to API access control.
- Improve the security risk mitigation delivered by API security controls by implementing API access control (to control which humans and machines can access APIs), alongside API discovery and API threat protection.
- Evaluate the quality of IAM tools by examining their capabilities for token exchange, access policy management, libraries and API gateway integrations.
- Reduce the incidence of API vulnerabilities by providing IAM training for developer and security staff, highlighting API access control.
- Protect services by deploying API access control enforcement points as close as possible to the service being protected. This will help enable defense in depth and supports a zero-trust approach.

Sample Vendors

Amazon Web Services; Authlete; Cloudentity; Curity; ForgeRock; Microsoft; Okta; Ping Identity

Gartner Recommended Reading

Magic Quadrant for Access Management

Critical Capabilities for Access Management

Architect a Modern API Access Control Strategy

API Security: What You Need to Do to Protect Your APIs

Critical Capabilities for Full Life Cycle API Management

API Developer Portals in Banking

Analysis By: Don Free

Benefit Rating: Moderate

Market Penetration: 20% to 50% of target audience

Maturity: Early mainstream

Definition:

API developer portals in banking provide the capacity to document, configure and manage usage of APIs for application-to-application communications.

Why This Is Important

- API developer portals provide an advanced approach to connect and manage APIs of partner solutions embedded within bank business processes. They're also used to maintain internal APIs.
- Accelerating internal developer portal usage and adoption is crucial to banks. These portals promote awareness and centralize documentation, which increases ease of use for internal developers and third parties.
- API developer portals are foundational for banks with open banking aspirations.

Business Impact

APIs play a key role in boosting banks' digital business initiatives, and API developer portals promote and optimize API consumption and performance through both functional and technical documentation and tools.

Drivers

- Transparency and self-service access are increasingly important characteristics for successful API development and usability.
- API resources are broadening within banks, and the distributed management of these software assets will intensify complexity, drive duplication and increase total cost of ownership (TCO); API developer portals help address these issues.
- The API developer portal is a foundational technology and a prerequisite for API marketplaces that support the pursuit of higher-order business models such as banking as a service.
- Internal communication and collaboration are fundamental attributes that promote successful API programs.
- API developer portals may also be adapted for usage by low-code platforms.

Obstacles

- Low internal developer portal adoption rates and decreased trust will promote siloed behaviors and erode innovation.
- Complex navigation and various versions of APIs diminish usage rates among developers.
- Assumptions that internal developer portals are intended only for developers will diminish value.
- Lack of a broader standards strategy and adoption by banks can lead to additional requirements for extended API transformation and corresponding development costs.
- A lack of oversight on API reusability inclusive of documentation and where APIs are used hinders standards implementation and increases the potential for higher technology debt.

User Recommendations

- Increase developer portal adoption and credibility by making developers successful by, for example, building skill sets and providing tools to drive efficiency.
- Drive user adoption through marketing efforts that recognize developers and showcase use cases that contribute to business value and have the potential to be reused.
- Include the developer portal as part of your overall application development strategy, and integrate it within application development centers (onshore and offshore).
- Use a “standards first” approach for applications and APIs such as the Financial Data Exchange (FDX) and the Banking Industry Architecture Network (BIAN).

Sample Vendors

Achieve Internet; Axway; Google; IBM; Pronovix; Salesforce

Gartner Recommended Reading

Leverage API Portals for Successful API Adoption

Open Banking

Analysis By: Alistair Newton

Benefit Rating: Transformational

Market Penetration: 20% to 50% of target audience

Maturity: Mature mainstream

Definition:

Open banking enables access to data and services across ecosystems of third-party financial service providers, banks and their customers via open APIs. Open banking initiatives are the formal embodiment and integration of open banking principles and technologies to deliver against a bank’s digital banking and technology strategy. Open banking initiatives may range from transformative through to simple regulatory compliance.

Why This Is Important

Many banks initially focused tactically on open banking technologies, addressing regulatory requirements as the priority. However, increasing numbers of banks view open banking as a means to innovate and integrate it into business strategy. Banks with transformative ambitions are using it to drive collaboration with fintechs, deliver product innovation and support agile business processes. Whether they enable regulatory compliance or support high levels of integration and innovation, open banking initiatives should be central to bank CIOs' thinking.

Business Impact

An open banking strategy will enable banks to position their ecosystem and API-focused technology investments in the context of the overarching business strategy, which:

- Will help CIOs and senior business executives align open banking strategies with digital banking initiatives, enterprise architectures, infrastructures and operations innovation.
- Is essential to create the platforms, marketplaces, app stores, and ecosystems necessary to create new value and revenue for the bank.

Drivers

Open banking initiatives take two distinct paths as either regulatory-led, which requires banks to open their customer data to third-party applications, or market-led, with individual or groups of banks focusing on open banking to drive digital transformation. Thus:

- Global adoption of open banking regulation is accelerating with multiple countries in the APAC region, the GCC region and across Central and South America.
- Regulatory-driven open banking is still seen by many banks as a challenge to their position and enabling fintech challengers at their expense.
- Initial open banking initiatives focused on the portability of payment data — now innovations in the payment context are growing in importance.
- Innovative open banking banks offer developer portals to enable access APIs to integrate banking products into third-party applications or services. Developer portals are now extremely common among those banks driving innovation through open banking.

- Transformative open banking can allow banks to create a marketplace that supports consumer, SMB and corporate customers, either by providing nonbanking services or allowing deep integration of the bank's own products into the back-office environment of the customer.
- Increasingly, Gartner is seeing this approach translated into wider banking-as-a-service and embedded finance strategies.
- Open banking strategies in some geographic markets where open banking initiatives are more developed will reach the Plateau of Productivity within two years. However, due to significant global variations in adoption, the creation of a well-defined open banking strategy will take two to five years to reach the Plateau of Productivity.

Obstacles

CIOs will struggle to support true digital transformation without a number of open banking initiatives:

- Recognizing the impact of disintermediation and regulatory changes, executives are looking to their CIOs for guidance on how to leverage technology to address these challenges.
- In response, IT leaders have begun to explore open banking strategies in partnership with the business, and while progress has accelerated, in many institutions, a more expansive view on the innovation opportunities would be beneficial.
- Transformational open banking strategies will challenge many deeply held beliefs and approaches in a bank. These include business capabilities (such as identifying and recruiting business ecosystem partners), people and culture (such as hiring talent with experience building and supporting APIs), information and technology (such as building a digital platform) and business ecosystems (such as turning competitors into customers).

User Recommendations

CIOs should create an executive strategy team composed of risk, compliance, marketing, IT and business leaders responsible for an open banking strategy, and:

- Identify the strategic context. Platform business can reduce banks to invisible, back-end utility service providers unless they change their business strategies. Doing nothing is not an option.

- Define future business objectives, goals and strategies, like, “We will compete by providing higher-quality tools and by sharing data, algorithms and transactions.”
- Create strategic principles, such as, “We will empower business ecosystems to create new value for our customers,” and, “We will expose business services to be shared internally and externally.”
- Identify measures of success and KPIs such as the conversion rate of customers to business ecosystem partners or percentage of SLAs met.
- Identify risks and issues, such as culture risk, ensure that the strategy is subject to constant review and sense-checking, and create a mitigation strategy.

Entering the Plateau

API Management PaaS

Analysis By: Mark O'Neill

Benefit Rating: Moderate

Market Penetration: 20% to 50% of target audience

Maturity: Mature mainstream

Definition:

API management PaaS (APIM PaaS) takes an on-demand approach to the delivery of API management by providing an alternative to the installation and management of API management software. APIM PaaS manages API access via provider-hosted API gateway services, and it may also include an API developer portal. In some cases, an on-premises API gateway option will be provided. APIM PaaS may be optimized to be used with PaaS services from the same vendor, such as function PaaS (fPaaS).

Why This Is Important

APIM PaaS takes full advantage of cloud benefits, such as autoscaling, resiliency and robust security. It also allows some vendors to offer per-API-call pricing. APIM PaaS may include the ability to deploy on-premises API gateways, to enable hybrid API management architecture with APIs on-premises and to offer cloud-based API management.

Business Impact

APIM PaaS allows costs to scale with the business value of APIs, reducing the impact of a large outlay as an API program grows. It enables APIs to be managed effectively when API traffic is unpredictable and potentially very large. APIM PaaS also brings business benefits when an APIM PaaS offering is provided as part of the PaaS platforms already in use by an organization, through unified procurement and billing.

Drivers

- APIM PaaS is driven by migration to and adoption of cloud platforms.
- An increasing number of available APIs and a growing volume of API interactions drive organizations to APIM PaaS for its high-scale quality of service, including throughput, high availability, and global access.

- SaaS adoption is also a driver, as organizations wish to use API management without needing to operate and maintain an API management software.
- Serverless computing, including fPaaS (also known as function as a service or FaaS), can act as a major driver for APIM PaaS. This is because fPaaS offerings can make use of API management on their associated cloud platforms. In some cases, they can automatically populate API gateways with endpoints so that fPaaS functions can be called via REST APIs.
- Since many organizations build APIs in the cloud, APIM PaaS is also increasingly used in hybrid and multicloud scenarios.
- Automation is also a driver for APIM PaaS. This is because APIM PaaS itself includes documented APIs in the API management platform. These APIs are used for tasks such as registering APIs. APIM PaaS typically can automatically register APIs built on the same platform.

Obstacles

- APIM PaaS tends to focus on runtime (API gateway) capabilities, with limited support for an API developer portal or other aspects of API management beyond API gateways.
- Network latency concerns impact the uptake of APIM PaaS for managing on-premises APIs. Even in a hybrid scenario, any requirement for the remote gateway to connect to the core platform introduces latency concerns.
- Data residency concerns, such as a storage of API payloads that may contain private information, are also an obstacle to the uptake of APIM PaaS for managing on-premises APIs.
- APIM PaaS can result in higher-than-expected costs as API traffic grows.
- APIM PaaS solutions from cloud hyperscalers are generally tied to their larger PaaS platforms, and are not portable for their use on other PaaS platforms.

User Recommendations

- Investigate the use of APIM PaaS to provide a cost-effective means of providing API management. If some or all of your APIs are on-premises, then investigate a hybrid option.
- For organizations migrating to the cloud, investigate hybrid APIM PaaS options.

- Compare the pricing of APIM PaaS vendors, since not all provide consumption-based pricing.
- Include API PaaS as part of your API strategy, since it can accelerate the time to market of your mission-critical digital initiatives.

Sample Vendors

Alibaba Cloud; Amazon Web Services; Google (Apigee); Huawei, IBM; Microsoft; Oracle; VMware

Gartner Recommended Reading

Magic Quadrant for Full Life Cycle API Management

Reference Model for API Management Solutions

Infographic: Decision Point for API and Service Implementation Architecture

Full Life Cycle API Management

Analysis By: Shameen Pillai

Benefit Rating: High

Market Penetration: More than 50% of target audience

Maturity: Early mainstream

Definition:

Full life cycle API management involves the planning, design, implementation, testing, publication, operation, consumption, versioning and retirement of APIs. API management tools help creating API ecosystems, publishing and operating APIs, and collecting analytics for monitoring and business value reporting. These capabilities are typically packaged as a combination of a developer portal, API gateway, API design, development and testing tools as well as policy management and analytics.

Why This Is Important

APIs are widely used as the primary choice to connect systems, applications and devices; integrate business partners, and build modular and composable software architectures. The use of APIs as digital products — monetized directly or indirectly — is also on the rise. Digital transformation initiatives have accelerated the need for creation, management, operations and security of APIs. They have also made full life cycle API management an essential foundational capability for every organization.

Business Impact

Full life cycle API management provides the framework and tools necessary to manage and govern APIs that are foundational elements of multiexperience applications, composable architectures and key enablers of digital transformations. It enables the creation of API products, which may be directly or indirectly monetized, while its security features serve to protect organizations from the business risk related to API breaches.

Drivers

- Organizations are facing an explosion of APIs, stemming from the need to connect systems, applications, devices and other businesses. The use of APIs in internal, external, B2B, private and public sharing of data is driving up the need to manage and govern APIs using full life cycle API management.
- APIs that package data, services and insights are increasingly being treated as products that are monetized and enable platform business models. Full life cycle API management provides the tooling to treat APIs as products.
- Digital transformation drives an increased use of APIs, which in turn increases the demand for full life cycle API management.
- Organizations looking for growth acceleration and business resilience are adopting API-based architectures to eliminate or modernize their legacy and monolithic applications.
- Developer mind share for APIs is growing. Newer approaches to event-based APIs, design innovations and modeling approaches — such as GraphQL — are driving interest in full life cycle API management, leading to more experimentation and growth.
- Cloud adoption and cloud-native architectural approaches to computing (including serverless computing) are increasing the use of APIs in software engineering architectures, especially in the context of microservices, service mesh and serverless.
- Greater awareness of and increasing significance to API security are driving organizations to take charge of discovering and managing APIs. This often starts with operational management of existing APIs.
- Regulated, industry-specific initiatives in the financial, insurance and healthcare industries continue to provide a steady demand. However, use of APIs is spreading across most industries (especially retail, technology and telecom, manufacturing), increasing the demand.
- Commoditization and widespread availability of API gateways as part of cloud services, security solutions and other bundled software applications are increasing the need for distributed API management that involves multiple gateways, including those from multiple vendors.
- Rising influence of generative AI and large language models in software engineering is likely to increase and reshape the need for APIs.

Obstacles

- A lack of commitment to adequate organizational governance processes hinders the adoption of full life cycle API management. This can be due to a lack of skills or know-how, or due to putting too much focus on bureaucratic approaches rather than federated and automated governance approaches.
- A lack of strategic focus on business value (quantifiable business growth or operational efficiency) and too much focus on technical use cases can disengage business users and sponsors. This is particularly apparent in cases where API programs fail to deliver the promised return on investment.
- Traditional, single-gateway approaches to API management no longer fit well with modern, distributed API management approaches.
- A partial or full set of embedded API management capabilities provided by vendors in other markets — such as application development, integration platforms, security solutions and B2B offerings — can partially meet the use cases for API management and shrink the market opportunities.

User Recommendations

- Use full life cycle API management to power your API strategy and address both technical and business requirements for APIs. Select offerings that can address both well beyond the first year.
- Treat APIs as products managed by API product managers in a federated API platform team. Adopt business metrics for API products.
- Choose a functionally broad API management solution that supports modern API trends, including microservices, multigateway and multicloud architectures. Ensure that the chosen solution covers the entire API life cycle.
- Use full life cycle API management to enable governance of all APIs, including third-party (private or public) APIs that you consume.
- Ask full life cycle API management vendors about their support for automation, especially for API validation, testing and devOps integration. Also ask about support for low-footprint and third-party API gateways.
- Full life cycle API management solutions are suitable for implementing a single-vendor strategy as opposed to a best-fit tooling approach for different API life stages. Ensure the choice aligns with your organization's goals.

Sample Vendors

Axway; Google; IBM; Kong; Microsoft; Salesforce (MuleSoft); Software AG

Gartner Recommended Reading

Magic Quadrant for Full Life Cycle API Management

The Evolving Role of the API Product Manager in Digital Product Management

How to Use KPIs to Measure the Business Value of APIs

API Security: What You Need to Do to Protect Your APIs

Reference Model for API Management Solutions

Appendixes

See the previous Hype Cycle: Hype Cycle for APIs, 2022

Hype Cycle Phases, Benefit Ratings and Maturity Levels

Table 2: Hype Cycle Phases

(Enlarged table in Appendix)

Phase	Definition
Innovation Trigger	A breakthrough, public demonstration, product launch or other event generates significant media and industry interest.
Peak of Inflated Expectations	During this phase of overenthusiasm and unrealistic projections, a flurry of well-publicized activity by technology leaders results in some successes, but more failures, as the innovation is pushed to its limits. The only enterprises making money are conference organizers and content publishers.
Trough of Disillusionment	Because the innovation does not live up to its overinflated expectations, it rapidly becomes unfashionable. Media interest wanes, except for a few cautionary tales.
Slope of Enlightenment	Focused experimentation and solid hard work by an increasingly diverse range of organizations lead to a true understanding of the innovation's applicability, risks and benefits. Commercial off-the-shelf methodologies and tools ease the development process.
Plateau of Productivity	The real-world benefits of the innovation are demonstrated and accepted. Tools and methodologies are increasingly stable as they enter their second and third generations. Growing numbers of organizations feel comfortable with the reduced level of risk; the rapid growth phase of adoption begins. Approximately 20% of the technology's target audience has adopted or is adopting the technology as it enters this phase.
Years to Mainstream Adoption	The time required for the innovation to reach the Plateau of Productivity.

Source: Gartner (July 2023)

Table 3: Benefit Ratings

Benefit Rating ↓	*Definition* ↓
Transformational	Enables new ways of doing business across industries that will result in major shifts in industry dynamics
High	Enables new ways of performing horizontal or vertical processes that will result in significantly increased revenue or cost savings for an enterprise
Moderate	Provides incremental improvements to established processes that will result in increased revenue or cost savings for an enterprise
Low	Slightly improves processes (for example, improved user experience) that will be difficult to translate into increased revenue or cost savings

Source: Gartner (July 2023)

Table 4: Maturity Levels

(Enlarged table in Appendix)

Maturity Levels ↓	*Status* ↓	*Products/Vendors* ↓
Embryonic	In labs	None
Emerging	Commercialization by vendors Pilots and deployments by industry leaders	First generation High price Much customization
Adolescent	Maturing technology capabilities and process understanding Uptake beyond early adopters	Second generation Less customization
Early mainstream	Proven technology Vendors, technology and adoption rapidly evolving	Third generation More out-of-box methodologies
Mature mainstream	Robust technology Not much evolution in vendors or technology	Several dominant vendors
Legacy	Not appropriate for new developments Cost of migration constraints replacement	Maintenance revenue focus
Obsolete	Rarely used	Used/resale market only

Source: Gartner (July 2023)

Evidence

The 2022 Gartner Software Engineering Leaders Role Survey was conducted to understand how organizations:

- Attract, hire and retain software engineering talent
- Improve and modernize developer skills
- Improve developer productivity
- Establish platform engineering teams
- Create platform teams
- Incorporate design into software engineering

The survey was conducted online from November through December 2022. In total, 300 respondents were interviewed from the United States. Qualifying organizations operated in multiple industries (excluding IT software and public sector) and reported enterprisewide revenue for fiscal year 2021 of at least $250 million or equivalent, with 60% reporting over $1 billion in revenue. Qualified participants were highly involved in managing software engineering/application development teams and the activities they perform.

Disclaimer: Results of this survey do not represent global findings or the market as a whole, but reflect the sentiments of the respondents and companies surveyed.

Document Revision History

Hype Cycle for APIs, 2022 - 10 August 2022

Hype Cycle for APIs and Business Ecosystems, 2021 - 27 July 2021

Hype Cycle for API and Business Ecosystems, 2020 - 7 August 2020

Hype Cycle for Business Ecosystems, 2019 - 31 July 2019

Hype Cycle for Business Ecosystems, 2018 - 6 August 2018

Hype Cycle for Business Ecosystems, 2017 - 21 July 2017

Recommended by the Authors

Some documents may not be available as part of your current Gartner subscription.

Understanding Gartner's Hype Cycles

Tool: Create Your Own Hype Cycle With Gartner's Hype Cycle Builder

Research Index: Everything You Should Do to Address API Security

Reference Model for API Management Solutions

Table 1: Priority Matrix for Hype Cycle for APIs

Benefit ↓	*Years to Mainstream Adoption*			
	Less Than 2 Years ↓	*2 - 5 Years* ↓	*5 - 10 Years* ↓	*More Than 10 Years* ↓
Transformational	Open Banking	API-Based Digital Commerce CSP Open APIs		
High	API Access Control API Testing Services Full Life Cycle API Management	API-Centric SaaS API Observability API Security Testing API Threat Protection Business Ecosystem Modeling Event-Driven Architecture Microservices	API Monitoring Federated API Gateways FHIR APIs Open APIs in Insurance	
Moderate	API Developer Portals in Banking API Management PaaS	API Industry Standards API Marketplaces Financial APIs GraphQL APIs	Event-Driven APIs	
Low			API Marketplaces in Supply Chain Planning	

Source: Gartner (July 2023)

Table 2: Hype Cycle Phases

Phase	*Definition*
Innovation Trigger	A breakthrough, public demonstration, product launch or other event generates significant media and industry interest.
Peak of Inflated Expectations	During this phase of overenthusiasm and unrealistic projections, a flurry of well-publicized activity by technology leaders results in some successes, but more failures, as the innovation is pushed to its limits. The only enterprises making money are conference organizers and content publishers.
Trough of Disillusionment	Because the innovation does not live up to its overinflated expectations, it rapidly becomes unfashionable. Media interest wanes, except for a few cautionary tales.
Slope of Enlightenment	Focused experimentation and solid hard work by an increasingly diverse range of organizations lead to a true understanding of the innovation's applicability, risks and benefits. Commercial off-the-shelf methodologies and tools ease the development process.
Plateau of Productivity	The real-world benefits of the innovation are demonstrated and accepted. Tools and methodologies are increasingly stable as they enter their second and third generations. Growing numbers of organizations feel comfortable with the reduced level of risk; the rapid growth phase of adoption begins. Approximately 20% of the technology's target audience has adopted or is adopting the technology as it enters this phase.
Years to Mainstream Adoption	The time required for the innovation to reach the Plateau of Productivity.

Phase	Definition

Source: Gartner (July 2023)

Table 3: Benefit Ratings

Benefit Rating	Definition
Transformational	Enables new ways of doing business across industries that will result in major shifts in industry dynamics
High	Enables new ways of performing horizontal or vertical processes that will result in significantly increased revenue or cost savings for an enterprise
Moderate	Provides incremental improvements to established processes that will result in increased revenue or cost savings for an enterprise
Low	Slightly improves processes (for example, improved user experience) that will be difficult to translate into increased revenue or cost savings

Source: Gartner (July 2023)

Table 4: Maturity Levels

Maturity Levels ↓	*Status* ↓	*Products/Vendors* ↓
Embryonic	In labs	None
Emerging	Commercialization by vendors Pilots and deployments by industry leaders	First generation High price Much customization
Adolescent	Maturing technology capabilities and process understanding Uptake beyond early adopters	Second generation Less customization
Early mainstream	Proven technology Vendors, technology and adoption rapidly evolving	Third generation More out-of-box methodologies
Mature mainstream	Robust technology Not much evolution in vendors or technology	Several dominant vendors
Legacy	Not appropriate for new developments Cost of migration constraints replacement	Maintenance revenue focus
Obsolete	Rarely used	Used/resale market only

Source: Gartner (July 2023)