Cycle de battage médiatique autour des API, 2024

2 juillet 2024 - ID G00809980 - 132 min de lecture

Par Mark O'Neill et John Santoro

Le monde des API est en pleine mutation. Les développeurs utilisent de plus en plus l'IA générative pour créer et utiliser des API. GraphQL, gRPC et AsyncAPI sont de plus en plus utilisés, tandis que les attaques de sécurité des API sont désormais courantes. Ce Hype Cycle met en lumière les principales tendances des API pour les responsables de l'ingénierie logicielle et leurs équipes.

Analyse

Ce que vous devez savoir

Ce document a été révisé le 10 juillet 2024. Le document que vous consultez est la version corrigée. Pour plus d'informations, consultez la page Corrections sur gartner.com.

L'utilisation des API continue de croître, de nombreuses organisations utilisant plusieurs types d'API. Dans l' enquête Gartner API Strategy Survey 2024, 82 % des répondants ont déclaré que leurs organisations utilisent des API en interne, tandis que 71 % utilisent également des API fournies par des tiers tels que des fournisseurs SaaS. Les API tierces largement utilisées incluent désormais les API d'IA générative (GenAI) de fournisseurs tels qu'OpenAI.

En plus de leur utilisation croissante, les API se diversifient également. Bien que REST domine toujours l'utilisation des API à 85 %, des alternatives telles qu'AsyncAPI (20 % d'adoption), GraphQL (19 % d'adoption) et gRPC (11 % d'adoption) sont en croissance [1]. L'utilisation croissante de ces alternatives entraîne la nécessité de choisir une infrastructure API qui prend en charge non seulement REST mais également d'autres formats d'API.

Un autre facteur à prendre en compte lors du choix d'une infrastructure API est que les organisations ont de plus en plus besoin d'une combinaison de différentes passerelles API, de portails API et d'outils spécialisés de test, d'observabilité et de surveillance des API . Ce cycle de battage médiatique reflète cette tendance au « dégroupage des plateformes » en incluant des entrées pour ces catégories de produits distinctes.

Les acquisitions sont un facteur constant dans le monde des API, notamment récemment :

- Boomi acquiert l'activité de gestion fédérée des API d'APIIDA

- Akamai Technologies va acquérir Noname Security
- Optic racheté par Atlassian

La sécurité reste le principal défi de la stratégie API .[1] Par conséquent, les tests de sécurité des API et la protection contre les menaces des API sont essentiels. La sécurité des API comprend l'exigence de découvrir les API afin de cartographier la surface d'attaque des API d'une organisation.

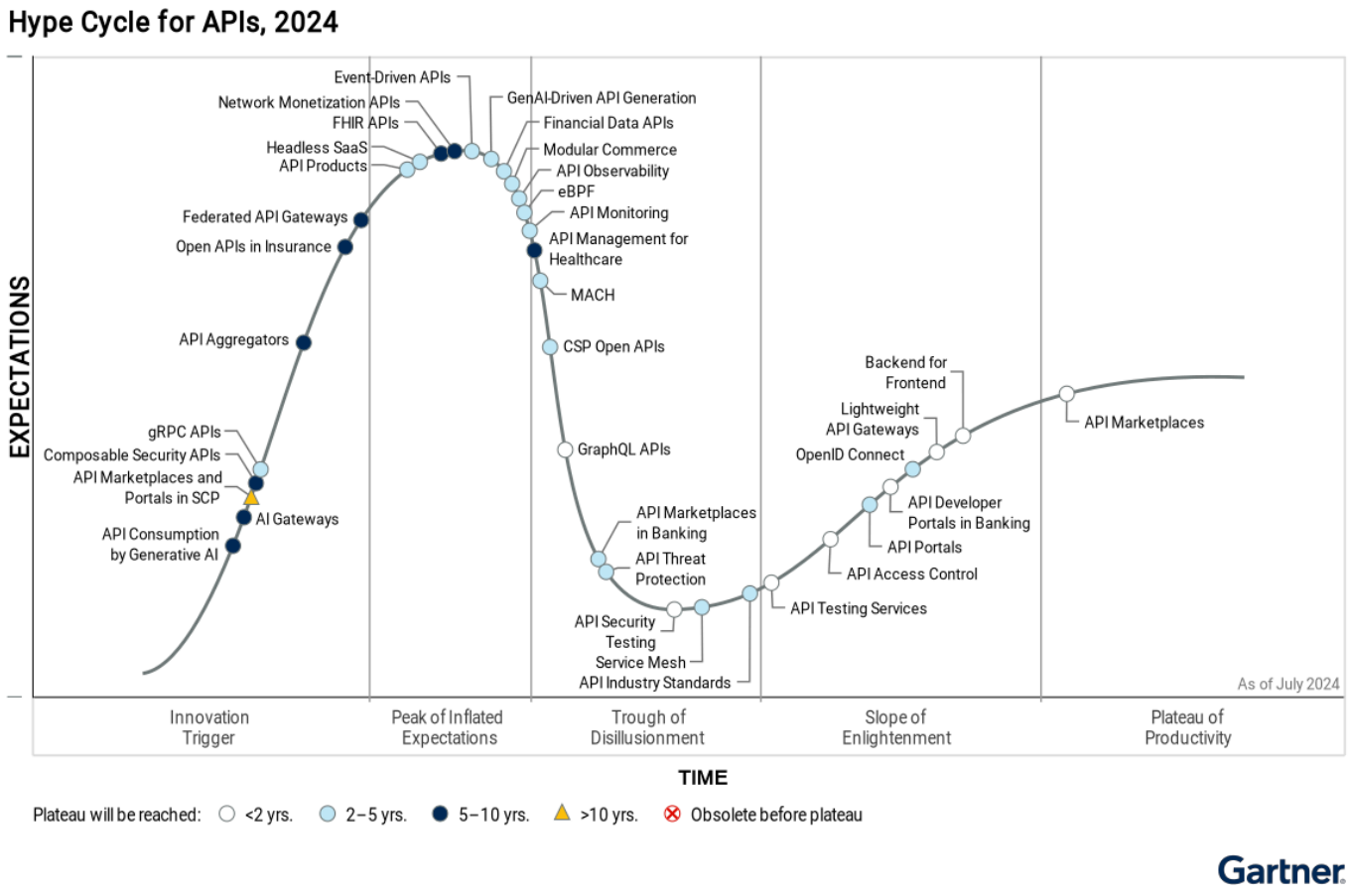
Enfin, les compétences en matière d'API sont plus importantes que jamais. Dans l'enquête Gartner Software Engineering Survey pour 2024, 57 % des responsables de l'ingénierie logicielle ont indiqué que la conception et le développement d'API sont une compétence très importante pour fournir des logiciels qui répondent aux besoins de l'entreprise, juste derrière la sécurité des applications (75 %).[2]

Le cycle du battage médiatique

Ce cycle de battage médiatique met en évidence les tendances API les plus importantes :

- Les développeurs utilisent de plus en plus d'outils tels que les assistants de code IA pour concevoir et créer des API (voir **le Guide d'innovation pour les assistants de code IA**). Dans le même temps, les grands modèles de langage (LLM) deviennent de grands consommateurs d'API. Par conséquent, la génération d'API pilotée par GenAI et la consommation d'API par GenAI sont toutes deux présentes dans le Hype Cycle de cette année.
- Les API sectorielles continuent de mûrir, certaines industries devançant d'autres. Cette tendance se reflète dans les entrées du Hype Cycle : les places de marché et portails d'API dans la planification de la chaîne d'approvisionnement (SCP), les API ouvertes dans l'assurance, les API FHIR (soins de santé), les API de données financières, le commerce modulaire, les API ouvertes des fournisseurs de services de communication (CSP) et les places de marché d'API dans le secteur bancaire.
- Les difficultés d'adoption de la sécurité des API et d'identification et de blocage des menaces API se reflètent dans le positionnement des tests de sécurité des API et de la protection contre les menaces API, qui sont désormais dans le creux de la désillusion.
- GraphQL et gRPC sont de plus en plus adoptés. GraphQL est plus largement adopté que gRPC, ce qui se reflète dans sa position dans le Hype Cycle. Cependant, GraphQL tombe dans le creux de la désillusion en raison de préoccupations concernant la complexité et la sécurité.
- Les produits API restent une tendance importante, proche du sommet du cycle de battage médiatique, avec un intérêt significatif de la part des praticiens des API.
- AsyncAPI et d'autres technologies d'API pilotées par événements telles que les webhooks suscitent un intérêt considérable de la part de la communauté, ce qui se reflète dans le positionnement des API pilotées par événements à proximité du sommet .

Figure 1. Cycle de battage médiatique pour les API, 2024



La matrice des priorités

Les nouvelles entrées dans la phase de déclenchement de l'innovation comprennent :

- API de sécurité composables, qui permettent aux organisations d'intégrer des fonctionnalités de sécurité dans leurs applications
- Agrégateurs d'API, reflétant une prise de conscience croissante de l'importance des agrégateurs d'API dans des secteurs tels que le secteur bancaire
- Consommation d'API par GenAI, car les LLM deviennent de grands consommateurs d'API

Les marchés et portails d'API dans SCP et les API de monétisation de réseau sont les plus éloignés de l'adoption générale, tous deux étant à plus de 10 ans du plateau de productivité. Cela signifie que les adoptants doivent aborder ces technologies avec prudence, en reconnaissant que le retour sur investissement peut être très lointain.

Dans le secteur de la santé, les API FHIR et la gestion des API pour les soins de santé bénéficient d'un taux d'avantages « élevé » et d'un délai d'adoption généralisé plus proche , ce qui signifie que les organisations de soins de santé qui adoptent ces technologies peuvent s'attendre à des avantages plus rapides.

Tableau 1 : Matrice de priorité pour les API, 2024

Avantage	Années avant l'adoption généralisée			
	Moins de 2 ans	2 - 5 ans	5 - 10 ans	Plus de 10 ans
Transformationnel		API ouvertes CSP Commerce modulaire	Consommation d'API par l'IA générative API de monétisation du réseau	
Haut	Contrôle d'accès API Tests de sécurité des API	Surveillance des API Observabilité des API Portails API Produits API Protection contre les menaces API SaaS sans tête Connexion OpenID	Gestion des API pour les soins de santé Passerelles API fédérées API FHIR API ouvertes dans le domaine de l'assurance	Marchés et portails API dans SCP

Avantage	Années avant l'adoption généralisée			
	Moins de 2 ans	2 - 5 ans	5 - 10 ans	Plus de 10 ans
Modéré	**Portails de développeurs d'API dans le secteur bancaire** **Marchés d'API** **Services de test d'API** **Backend pour Frontend** **API GraphQL**	**Normes de l'industrie API** **Les marchés API dans le secteur bancaire** **eBPF** **API pilotées par événements** **API de données financières** **Génération d'API pilotée par GenAI** **API gRPC**	**Passerelles d'IA** **Agrégateurs d'API** **API de sécurité composables**	
Faible	**Passerelles API légères**	**MACH** **Maillage de services**		

Source : Gartner (juillet 2024)

Hors du cycle du battage médiatique

La gestion des API PaaS, la gestion du cycle de vie complet des API et l'open banking ont tous dépassé le plateau de productivité et ne sont donc plus dans le cycle de battage médiatique des API.

À la hausse

Consommation d'API par l'IA générative

Analyse par : Mark O'Neill, Arun Batchu

Évaluation des avantages : Transformationnel

Pénétration du marché : 1 à 5 % du public cible

Maturité : Émergente

Définition:

La consommation d'API par l'IA générative (GenAI) implique que de grands modèles de langage et d'autres technologies GenAI consomment des données et accèdent aux fonctionnalités de l'application via des API. Cela implique généralement que la technologie d'IA consomme la définition d'API, via une définition OpenAPI, et appelle l'API.

Pourquoi c'est important

Dans son OpenAPI Moonwalk 2024, la Fondation OpenAPI a identifié « l'émergence d'un nouveau type de consommateur d'API : l'IA générative ». Les technologies GenAI vont croître en tant qu'utilisateurs d'API. Cela est dû à des technologies telles que les agents IA et les flux d'agents. En outre, les développeurs utilisent de plus en plus d'outils tels que les assistants de codage IA. Ces outils aident les développeurs à découvrir des API, à appeler des API et à générer du code, des tests et de la documentation .

Impact sur les entreprises

Le développement de logiciels stimule l'innovation et la croissance des entreprises. Les organisations doivent continuellement améliorer les processus de livraison de logiciels et mettre en œuvre des pratiques contemporaines pour attirer les développeurs. Gartner prédit que les assistants de codage IA augmenteront la productivité des développeurs de 36 % d'ici 2028 (pour en savoir plus, consultez **Comment communiquer la valeur des assistants de codage IA**). L'utilisation de GenAI pour aider les développeurs à utiliser les API dans les applications a un impact considérable sur la productivité des développeurs et améliore l'expérience de développement.

Conducteurs

- Les logiciels dépendent de plus en plus des API. L'une des utilisations intéressantes de GenAI est d'aider les développeurs à découvrir les API et à coder pour effectuer des requêtes API.
- Les outils de développement logiciel et les environnements de développement intégrés ajoutent rapidement des fonctionnalités GenAI à leurs produits. Les fonctionnalités spécifiques prenant en charge la consommation d'API sont un cas d'utilisation courant de GenAI.
- Les tests API générés par GenAI améliorent considérablement le délai de livraison des applications et contribuent à garantir une couverture complète des tests.
- La documentation générée par GenAI sur la consommation d'API dans une application fournit des informations et un contexte plus riches sur l' utilisation des API par l'application .

- La conception d'API selon les spécifications de l'industrie, en particulier OpenAPI, est essentielle pour la consommation par les modèles d'IA.

Obstacles

- Lorsque GenAI utilise des API, la sécurité doit être garantie. Cela peut nécessiter l'utilisation d'une infrastructure de sécurité API, telle qu'une passerelle API
- Lorsque les applications GenAI consomment des API, des dépendances sont créées qui peuvent ajouter de la latence.

Recommandations pour les utilisateurs

- La consommation d'API par GenAI dépend de la qualité des API et des définitions d'API de l'organisation. Mettez en œuvre une gestion des API qui garantit que votre organisation produit des API cohérentes et bien gérées.
- Choisissez une solution (comme la génération augmentée par récupération) pour rendre l' inventaire, les normes, les exemples et la documentation des API de l'organisation accessibles à la technologie GenAI sélectionnée. Cela permet aux réponses GenAI (telles que les API suggérées et le code pour invoquer les API) de refléter les API de l'entreprise et de se conformer à sa gouvernance des API.
- Conseiller et équiper les développeurs pour qu'ils puissent jouer des rôles plus stratégiques. Les développeurs doivent être capables de perspicacité et d'intuition pour examiner rapidement mais efficacement la sélection des API par GenAI et le codage suggéré par GenAI pour invoquer les API.
- Implémentez une sécurité et une protection API renforcées qui surveillent toutes les demandes d'API, y compris les demandes d'API provenant du code créé par GenAI, pour détecter toute activité API non conforme et suspecte.

Fournisseurs d'échantillons

LangChain ; LlamaIndex ; Tyk Technologies (Montag .AI)

Lectures recommandées par Gartner

Devenir une organisation axée sur l'IA : 5 phases critiques d'adoption de l'IA

Passerelles d'IA

Analyse par : Mark O'Neill, Andrew Humphreys

Évaluation des avantages : Modérée

Pénétration du marché : Moins de 1 % du public cible

Maturité : Embryonnaire

Définition:

Les passerelles d'IA, également appelées routeurs LLM ou passerelles multi-LLM, gèrent et sécurisent les connexions aux fournisseurs d'IA. Certaines passerelles d'IA sont basées sur des produits de passerelle API établis, car les API REST sont le principal mécanisme de connexion aux fournisseurs d'IA, tandis que d'autres sont conçues à cet effet. Les responsables de l'ingénierie logicielle peuvent utiliser les passerelles d'IA pour appliquer la sécurité, comme la protection des clés API émises par les fournisseurs d'IA, le routage multi-LLM, la visibilité des coûts et l'analyse de la confidentialité des données à l'utilisation de l'IA par leur organisation.

Pourquoi c'est important

À mesure que le volume et l'ampleur de l'IA générative et d'autres types de projets d'IA ont augmenté, les organisations ont besoin d'un plus grand contrôle sur leur utilisation des fournisseurs d'IA . Les passerelles d'IA assurent la gestion du trafic d'exécution entre l'organisation et ses fournisseurs d'IA. Les passerelles peuvent aider à mettre en œuvre et à gérer les contrôles de politique basés sur des invites, à suivre l'utilisation et les coûts des services d'IA , à acheminer les données sur plusieurs LLM et à gérer l'accès aux abonnements d'IA, y compris la protection des clés API émises par les fournisseurs d'IA.

Impact sur les entreprises

Les passerelles d'IA gèrent les interactions entre les applications et les modèles d'IA en assurant la sécurité, la gouvernance, l'observabilité et la gestion des coûts. Cela réduit le risque de coûts inattendus encourus par les fournisseurs d'IA et empêche que les données privées du trafic API soient compromises ou utilisées à mauvais escient. Une passerelle d'IA peut également permettre à une organisation de gérer l'utilisation de plusieurs modèles d'IA, provenant de différents fournisseurs, plutôt que de concentrer l'organisation sur un seul fournisseur.

Conducteurs

- **Contrôler les coûts d'accès :** les modèles de tarification des services d'IA ont tendance à être basés sur l'utilisation, ce qui présente un risque pour les entreprises car les coûts associés à l'utilisation de ces services peuvent augmenter rapidement. Les organisations cherchent à contrôler les coûts d'accès, et les passerelles d'IA peuvent les aider en mettant en cache les réponses pour limiter les appels en double et en suivant et en contrôlant l'accès aux services.
- **Activer la visibilité de l'utilisation des services d'IA :** les passerelles d'IA offrent un moyen d'obtenir une meilleure visibilité sur l'utilisation des API d'IA dans l'ensemble de l'organisation en exploitant les fonctionnalités d'observabilité et d'analyse de la gestion des API pour l'utilisation des API.
- **Optimisez l'accès aux moteurs d'IA :** les passerelles d'IA peuvent être configurées pour fournir une API unique devant plusieurs fournisseurs d'IA différents, tels que plusieurs LLM. Cela signifie que les développeurs peuvent accéder à plusieurs services d'IA à l'aide de la même API.
- **Renforcez la sécurité des interactions avec l'IA :** la gestion de la confiance, des risques et de la sécurité de l'IA (AI TRiSM) garantit la gouvernance, la fiabilité, l'équité, la fiabilité, la robustesse, l'efficacité et la protection des données du modèle d'IA. Les passerelles d'IA traitent des aspects de l'AI TRiSM, notamment la gouvernance et la fiabilité du modèle. En particulier, la

protection des clés API émises par les fournisseurs d'IA est essentielle. Si un attaquant parvient à accéder aux clés API d'une organisation pour un fournisseur d'IA, il peut accéder à des données privées et générer des factures d'utilisation élevées. Les passerelles d'IA peuvent être utilisées pour protéger les clés API émises par les fournisseurs d'IA.

Obstacles

- **Les passerelles d'IA sont nouvelles et n'ont pas encore fait leurs preuves.** Il est donc indispensable de procéder à une évaluation approfondie avant de s'engager auprès d'un fournisseur.
- **La latence** est une préoccupation pour tout intermédiaire, en particulier dans le domaine de l'IA, où les utilisateurs finaux peuvent remarquer un retard dans le temps de réponse. Contrairement à la latence traditionnelle des API, les mesures importantes ici sont le temps d'obtention du premier jeton et le jeton par seconde (ou le débit du jeton). Une passerelle d'IA peut aider les utilisateurs à optimiser ces deux mesures.
- **Un point de défaillance unique** constitue également un problème si une passerelle d'IA est déployée sans redondance en place.
- **La mise en cache des réponses des API d'IA est problématique et complexe,** car différents modèles d'IA peuvent utiliser des invites différentes. Il est difficile de déterminer si les requêtes en langage naturel demandent les mêmes informations , en particulier lorsqu'il s'agit d'une grande fenêtre de contexte.

Recommandations pour les utilisateurs

- Procédez à une évaluation approfondie d'une passerelle d'IA, car il s'agit d'une nouvelle catégorie avec des produits largement non éprouvés.
- Pour des scénarios plus simples tels que la limitation du débit des API d'IA, évaluez les capacités de votre passerelle API existante à agir comme une passerelle d'IA et à prendre en charge vos exigences.
- Demandez à vos fournisseurs d'IA s'ils peuvent fournir des fonctionnalités similaires, ce qui supprimerait la nécessité d'une passerelle d'IA distincte .

Fournisseurs d'échantillons

Aguru; Cloudflare; IBM; Kong; Lunar .dev; Martian; Portkey; Radiant

Lectures recommandées par Gartner

L'IA pousse les fournisseurs de gestion d'API à repenser leurs produits

Principales tendances technologiques stratégiques pour 2024 : gestion de la confiance, des risques et de la sécurité dans l'IA

Guide du marché des passerelles API

[Comment calculer la valeur commerciale et le coût des cas d'utilisation de l'IA générative]

[10 bonnes pratiques pour optimiser les coûts de l'IA générative]

Marchés et portails API dans SCP

Analyse par : Jan Snoeckx, Pia Orup Lund, Tim Payne

Évaluation des avantages : élevée

Pénétration du marché : Moins de 1 % du public cible

Maturité : Émergente

Définition:

Les marchés et portails d'API permettent aux consommateurs d'API (principalement les développeurs (c'est-à-dire les fournisseurs et les utilisateurs finaux)) de découvrir les API. Ils simplifient la recherche d'API et aident à expliquer leur fonctionnalité, afin que les développeurs puissent décider s'ils veulent les utiliser et, dans certains cas, permettre l'achat d'un accès à celles-ci. Un marché d'API donne accès aux API de plusieurs entreprises différentes, tandis qu'un portail d'API ne fournit que les API d'une seule entreprise.

Pourquoi c'est important

Les places de marché et les portails d'API permettent aux organisations de faire connaître leurs API. Ils sont généralement associés à des places de marché externes qui partagent des API avec une communauté de développeurs et permettent aux partenaires de mettre en œuvre des solutions à l'aide des API. Cependant, la plupart des API sont destinées à être utilisées au sein d'une organisation. Les places de marché et les portails peuvent donc également être internes. Ils facilitent la recherche d'API en interne, contribuant ainsi à un partage plus large des capacités entre les différentes unités commerciales et équipes de développement.

Impact sur les entreprises

Les places de marché et portails d'API dans les environnements de solutions de planification de la chaîne logistique (SCP) permettent aux utilisateurs d'étendre les capacités de leurs principales plateformes de planification. Ils peuvent stimuler l'utilisation des API, augmenter l'extensibilité, permettre l'accès aux données externes et, par extension, augmenter l'impact commercial. Les consommateurs d'API peuvent utiliser les places de marché et les portails pour simplifier la découverte et la comparaison des API lorsqu'ils recherchent des fonctionnalités spécifiques, mais n'ont pas sélectionné l' API qu'ils prévoient d'utiliser.

Conducteurs

- Le nombre d'API dans une organisation augmente, ce qui oblige les développeurs à découvrir plus facilement les API et les services disponibles.
- Les activités composables, y compris les SCP composables , s'appuient sur des marchés et des portails d'API pour partager des API et des fonctionnalités commerciales packagées .

- L' utilisation accrue de plateformes low-code, de plateformes d'intégration, d'automatisation des processus robotisés (RPA) et d'outils d'analyse permet un développement plus citoyen, en utilisant des API qui peuvent provenir de marchés et de portails d'API.

- Les nouvelles plateformes open source (par exemple, Backstage de Spotify) , ainsi que les produits commerciaux (par exemple, Cortex et Port), favorisent la création de marchés d'API internes dans le cadre de pôles de développement plus vastes.

Obstacles

- Peu de fournisseurs de logiciels SCP proposent un environnement ressemblant à une place de marché ou à un portail d'API. Certains construisent des écosystèmes de partenaires, mais rares sont ceux qui ont partagé la vision d'établir une place de marché d'API publique.

- Les marchés d'API publics qui fournissent un répertoire public d'API provenant de plusieurs fournisseurs ont généralement eu des résultats décevants , car les développeurs s'adressent généralement directement aux fournisseurs d'API pour s'inscrire aux API. Par conséquent , les fournisseurs SCP peuvent hésiter à lancer un marché d'API public. Cependant, les portails d'API internes ont eu plus de succès pour les fournisseurs d'API , car ils permettent aux développeurs de partager des API entre plusieurs équipes. Le temps nous dira si le même constat s'applique aux fournisseurs d'API SCP.

- Les portails API fournis dans le cadre des plateformes de gestion d'API sont généralement basiques, ce qui nécessite une personnalisation importante pour créer un marché d'API orienté client.

Recommandations pour les utilisateurs

Les fournisseurs d'API SCP doivent :

- Créez une place de marché ou un portail d'API interne axé sur les besoins des ingénieurs logiciels pour partager des API au sein de l'organisation, dans le cadre d'un portail de développeurs interne.

- Gérez les attentes des principales parties prenantes de l'entreprise en vous assurant qu'elles savent que les résultats du placement d'API sur les marchés d'API publics sont souvent décevants.

- Examinez les conditions de facturation pour déterminer ce qui revient au fournisseur de marché lorsque vous envisagez des marchés d'API commerciaux.

- Établissez en amont un modèle commercial (par exemple, via des frais d'inscription et/ou un partage des revenus) et un processus de gouvernance clair pour l'intégration d'API tierces, s'ils prévoient de créer leurs propres marchés d'API.

Les consommateurs de l'API SCP doivent :

- Assurez-vous qu'ils utilisent des API provenant de marchés et de fournisseurs d'API de confiance, en examinant attentivement les accords d'utilisation, les conditions de licence et de facturation.
- Recherchez si la consommation d'une API directement auprès d' un fournisseur d'API offre de meilleurs tarifs ou conditions d'utilisation que sa consommation via une place de marché ou un portail.

Lectures recommandées par Gartner

Innovation Insight pour les portails de développeurs internes

Guide du marché pour les plateformes de portail API

Modèle de référence pour les solutions de gestion des API

API de sécurité composables

Analyse par : Mark O'Neill, Manjunath Bhat

Évaluation des avantages : Modérée

Pénétration du marché : 1 à 5 % du public cible

Maturité : Émergente

Définition:

Les API de sécurité composables sont des fonctionnalités de sécurité telles que les coffres-forts de confidentialité, les services d'authentification, les services de chiffrement et les services de signature numérique fournis par les fournisseurs et généralement accessibles via des API. Les développeurs peuvent utiliser des API de sécurité composables pour intégrer des fonctionnalités de sécurité dans leurs applications. Les services de sécurité peuvent également être appelés à partir de plateformes de développement populaires.

Pourquoi c'est important

Les API de sécurité composables permettent aux développeurs d'intégrer des fonctionnalités de sécurité dans les applications à l'aide d'API. Étant donné qu'ils sont fournis par des API, ces services peuvent être utilisés sans nécessiter l'utilisation d'une plateforme de sécurité plus large. Les API de sécurité composables sont un élément important de la sécurité des applications personnalisées.

Impact sur les entreprises

- Les API de sécurité composables permettent aux développeurs de sélectionner et d'implémenter des fonctionnalités de sécurité dans les applications qu'ils créent. Cela incite les développeurs à devenir des sélectionneurs de technologies clés.
- L'utilisation d'API de sécurité composables crée des dépendances, qui ont un impact sur l'entreprise si le service devient indisponible ou si le fournisseur subit une faille de sécurité.

- Les API de sécurité composables permettent une meilleure relation entre les équipes de développement et de sécurité, ces dernières évaluant et approuvant les services. Les équipes d'ingénierie de la plateforme peuvent également prendre en charge ces services de sécurité.
- L'adoption d'une API de sécurité composable ciblée est généralement moins coûteuse que l'adoption d'une plateforme de sécurité plus grande.
- Les erreurs de configuration des mécanismes de sécurité sont une cause fréquente d'exposition pour les organisations. L'utilisation d'API de sécurité composables, fournies par un fournisseur spécialisé, peut réduire ce risque.

Conducteurs

- Les API sont désormais un élément essentiel du développement logiciel. Les développeurs s'attendent à ce que les services fournis par les API soient réutilisables dans le cadre d'une approche « API-first ».
- La standardisation des composants logiciels est une étape recommandée lors de la mise en place d'un programme de sécurité des applications organisationnelles. Cela favorise une approche « sécurisée par défaut ».
- Les développeurs ne sont généralement pas des experts en sécurité. L'utilisation d'API de sécurité composables leur permet d'inclure des fonctionnalités de sécurité dans les applications sans avoir à créer elles-mêmes les fonctionnalités. Par conséquent, l'application résultante est créée plus rapidement et de manière plus sûre que si elle était entièrement personnalisée.
- L'évolution vers des services composables en général stimule le marché des services de sécurité fournis sous forme d'API.
- Les développeurs ne sont plus seulement des influenceurs dans le choix des technologies ou de l'architecture, et ils sont désormais souvent à l'origine de la prise de décision et de la sélection des fournisseurs (pour en savoir plus, voir **L'influence omniprésente du développeur sur l'achat de technologies**). À leur tour, les fournisseurs ciblent de plus en plus les développeurs dans le cadre d'un mouvement de vente axé sur la croissance des produits. Cela stimule la croissance des produits axés sur les développeurs.

Obstacles

- L'utilisation d'API de sécurité composables comporte un risque de dépendances supplémentaires, une complexité architecturale et un risque de pannes ou le risque que le fournisseur de l'API subisse une faille de sécurité .
- Les API peuvent être difficiles à échanger ou à remplacer, ce qui crée des difficultés si les API de sécurité composables doivent être remplacées par une autre . Ce verrouillage potentiel constitue un obstacle majeur.

- Lorsque la sécurité est déjà intégrée à une plateforme ou peut être créée à l'aide d'assistants de codage, l'utilisation de services de sécurité distincts peut ne pas être nécessaire.

Recommandations pour les utilisateurs

- Évaluez l'utilisation d'API de sécurité composables pour les cas d'utilisation, notamment les coffres-forts, le chiffrement, l'analyse de fichiers et l'authentification.
- Élaborez un plan d'urgence si l'API devient indisponible ou si le fournisseur subit une violation.
- Utilisez des solutions de gestion des API pour surveiller la disponibilité et l'utilisation des API dont dépend votre organisation, y compris les API de sécurité composables. Une panne d'un service d'authentification, par exemple, signifie que les clients ou les employés ne peuvent pas se connecter aux applications.

Fournisseurs d'échantillons

BoxyHQ ; Evervault ; HashiCorp ; Okta (Auth0); Pangée ; Flux céleste

Lectures recommandées par Gartner

Structurer les outils et pratiques de sécurité des applications pour DevSecOps

Comment protéger vos applications cloud natives en production

API gRPC

Analyse par : Kevin Matheny

Évaluation des avantages : Modérée

Pénétration du marché : 5 à 20 % du public cible

Maturité : Adolescente

Définition:

gRPC est une norme pour les API d'appel de procédure à distance (RPC) à hautes performances et à faible latence avec des charges utiles binaires définies à l'aide de tampons de protocole envoyés via HTTP/2. Originaire de Google, qui l'a rendu open source en 2015, gRPC a progressivement été adopté, principalement pour les communications internes de service à service.

Pourquoi c'est important

Les architectures cloud natives modernes telles que les microservices s'appuient sur des API. Cependant, les API REST, la norme de facto, ne sont pas bien adaptées à la communication interservices à faible latence. gRPC optimise les performances élevées et la faible latence, et son typage fort permet de vérifier la sérialisation et la désérialisation des messages au moment de la compilation. gRPC prend en charge les communications synchrones et asynchrones, y compris les requêtes/réponses unaires et les RPC en streaming monodirectionnel et bidirectionnel .

Impact sur les entreprises

Les hautes performances et la faible latence de gRPC permettent aux équipes d'ingénierie logicielle de créer des applications rapides et efficaces , et son typage fort améliore la qualité des interfaces. Gartner constate une augmentation progressive de l'utilisation de gRPC pour les interfaces de communication de service à service. gRPC est principalement utilisé pour la communication au sein des domaines d'application plutôt que pour fournir des API réutilisables en interne ou en externe, pour lesquelles REST reste dominant et GraphQL gagne en popularité.

Conducteurs

- Les paradigmes d'architecture d'application modernes s'appuient sur des API pour la communication de service à service.
- Les API REST, la forme d'API la plus répandue, sont puissantes et flexibles. Cependant, elles ont une charge relativement élevée car la charge utile comprend des informations structurelles sur le contenu.
- gRPC optimise pour une faible latence et un débit élevé .

Obstacles

- gRPC diffère des API REST et GraphQL en utilisant une structure de charge utile binaire plutôt que la lisibilité humaine et l'omniprésence de JSON, et en nécessitant par défaut le protocole HTTP/2 plus performant .
- L'écosystème d'outils et de communautés entourant gRPC est plus petit que ceux de REST et GraphQL, ce qui limite la disponibilité des compétences et des outils. La prise en charge de gRPC dans les produits de gestion et d'intégration d'API est limitée.
- L'utilisation de gRPC nécessite que les équipes génèrent le code de sérialisation et de désérialisation des tampons de protocole (Protobuf) à partir du fichier IDL (Interface Description Language) côté serveur et côté consommateur. Les consommateurs ont besoin soit d'une copie du client généré spécifique au langage, soit d'une copie de l'IDL. Les consommateurs qui génèrent leur propre code de sérialisation/désérialisation à partir de l'IDL doivent utiliser des outils compatibles et risquent de commettre des erreurs lors de la génération.
- Les différences avec les autres formats d'API optimisent les performances de gRPC au détriment d'une certaine flexibilité et d'une certaine ubiquité.

Recommandations pour les utilisateurs

- Sélectionnez gRPC pour les API internes non publiées des services et des microservices afin de prendre en charge une communication interservices hautement performante et fortement typée.
- Évitez d'utiliser gRPC pour les API publiées en externe ou en interne et axées sur l'intégration. Utilisez plutôt REST ou GraphQL pour ces API.

Fournisseurs d'échantillons

F5 (NGINX); Google; Kong; Microsoft; Postman; Solo.io

Lectures recommandées par Gartner

Choisir un format d'API : REST avec OAS, GraphQL, gRPC ou AsyncAPI

Agrégateurs d'API

Analyse par : Mark O'Neill

Évaluation des avantages : Modérée

Pénétration du marché : 1 à 5 % du public cible

Maturité : Émergente

Définition:

Les agrégateurs d'API simplifient le développement d'applications en fournissant aux développeurs une API unique pour accéder à plusieurs systèmes ou organisations. Ils fournissent une API unique devant plusieurs services différents de différents fournisseurs, chacun pouvant avoir son propre ensemble d'API. Des agrégateurs d'API spécifiques à un secteur existent dans de nombreux secteurs, tels que la banque et la santé.

Pourquoi c'est important

- L'intégration avec plusieurs API représente un défi pour les développeurs en termes de coût et de complexité. Les agrégateurs d'API sont devenus populaires car ils fournissent des API uniques, qui s'interfacent avec plusieurs systèmes, tels que plusieurs banques ou systèmes de paie. Cela permet de gagner du temps de développement.
- Les agrégateurs d'API sont de plus en plus courants dans les secteurs présentant une certaine banalisation, tels que la banque, l'assurance ou la messagerie SMS.

Impact sur les entreprises

- Les agrégateurs d'API réduisent les délais de mise sur le marché lors de la création d'applications, car il est plus rapide d'utiliser une seule API que de l'intégrer directement à de nombreux systèmes utilisant différentes API. Cependant, le verrouillage de l'agrégateur d'API lui-même constitue un risque, comme l'arrêt ou les pannes d'un agrégateur d'API.
- Les agrégateurs d'API permettent aux fournisseurs d'applications de commercialiser leurs offres. Pour un fournisseur d'applications sans API, l'agrégateur d'API répond à ce besoin en fournissant des API que les clients peuvent utiliser. Cela peut être un moyen peu coûteux de fournir des API aux clients ou aux partenaires. Cependant, les agrégateurs d'API apportent une couche de désintermédiation.

Conducteurs

- Les organisations qui utilisent plusieurs systèmes similaires, en raison de facteurs tels que des acquisitions ou des différences géographiques au sein d'une organisation multinationale, sont susceptibles d'utiliser des agrégateurs d'API. Par exemple, une organisation qui souhaite s'intégrer à plusieurs systèmes RH différents peut utiliser un agrégateur d'API pour utiliser une seule API dans son application afin de se connecter aux différents systèmes RH.

- Les start-ups ont souvent besoin de pouvoir se connecter à de nombreux systèmes similaires. Par exemple, un produit de services financiers peut nécessiter de se connecter à de nombreuses banques, ou une application de voyage peut nécessiter de pouvoir se connecter à de nombreuses compagnies aériennes. Ce marché est un moteur pour les agrégateurs d'API.

Obstacles

- Les agrégateurs d'API présentent des risques en raison d'une dépendance à l'égard de l'agrégateur d'API lui-même.

- Les options pour les agrégateurs d'API sont limitées à certains secteurs, tels que la finance ou la santé, et à certains groupes de types d'applications SaaS, tels que les applications SaaS RH et CRM. Pour d'autres secteurs ou types d'applications, les agrégateurs d'API peuvent ne pas être disponibles.

Recommandations pour les utilisateurs

- Étudiez les agrégateurs d'API comme alternative à l'intégration avec plusieurs systèmes différents, mais similaires.

- Communiquez les risques liés à la dépendance à un agrégateur d'API comme point d'interaction unique avec plusieurs systèmes. Ces risques incluent des changements de prix inattendus ou des pannes.

- Interrogez les agrégateurs d'API sur leur stratégie de tarification. Est-elle basée sur le trafic, le nombre de connexions ou sur un abonnement à accès forfaitaire ?

Fournisseurs d'échantillons

Argyle; Finch; Tricot ; Fusionner; Plaid; TrueLayer

Lectures recommandées par Gartner

Comment mettre en œuvre avec succès l'intégration API-First
API ouvertes dans le domaine de l'assurance

Analyse par : Sham Gill

Évaluation des avantages : élevée

Pénétration du marché : Moins de 1 % du public cible

Maturité : Embryonnaire

Définition:

Les API ouvertes sont des interfaces de programmation d'applications publiées pour une utilisation externe par des tiers et des applications. Les API ouvertes d'assurance permettent aux développeurs des partenaires de distribution d'utiliser des portails en libre-service pour s'inscrire et accéder aux produits et services d'assurance.

Pourquoi c'est important

Les initiatives « ouvertes » ouvrent la voie à de nouvelles possibilités de disruption grâce à de nouveaux produits, de nouvelles opportunités de distribution et d'innovation. Les API ouvertes fournissent la technologie qui les fait fonctionner. Il est important que les DSI des compagnies d'assurance reconnaissent que les API ouvertes ne sont pas seulement une question de technologie. Elles permettent la connectivité entre la compagnie d'assurance et ses partenaires de l'écosystème externe. À ce titre, elles doivent être considérées comme des produits réutilisables à part entière, avec leur propre valeur commerciale intrinsèque .

Impact sur les entreprises

La publication de leurs propres API ouvertes et l'utilisation des API ouvertes de leurs partenaires peuvent accélérer la transformation numérique dans le secteur de l'assurance. Les API ouvertes peuvent étendre la portée d'un assureur à un public plus large et favoriser l'innovation et la transformation en offrant un échange de valeur fluide avec les partenaires de l'écosystème commercial. Elles permettent également une livraison plus rapide de nouveaux produits, services et modèles commerciaux, et permettent une monétisation directe et indirecte des API .

Conducteurs

- Le développement d'API internes ou privées pour des projets et des cas d'utilisation spécifiques peut être adapté au maintien du statu quo et sera prédominant dans un avenir proche. Mais elles ne suffiront pas aux assureurs qui prévoient d'adopter les écosystèmes commerciaux comme élément clé de leurs stratégies de transformation numérique de l'assurance.
- Les nouveaux modèles commerciaux exigeront une plus grande connectivité avec les partenaires de l'écosystème commercial à mesure que les assureurs innovants passeront des plateformes numériques aux plateformes commerciales.
- Les API ouvertes permettent aux assureurs d'atteindre leurs objectifs commerciaux :
  - Créez de nouveaux flux de revenus grâce à de nouveaux modèles commerciaux, produits et services distribués par des partenaires. L'assurance intégrée est un exemple où les API ouvertes/externes jouent un rôle essentiel en offrant aux partenaires de distribution un accès facile aux produits d'assurance.
  - Mieux répondre aux besoins d'évolution des attentes des consommateurs en matière d'accès numérique à l'assurance et d'accès et de contrôle des données personnelles.
  - Optimisez les processus en réduisant les frictions entre les partenaires et les clients. Par exemple, en permettant aux données de sinistres, aux images et aux analyses capturées lors du premier avis de sinistre (FNOL) d'être échangées de manière transparente avec les

partenaires de réparation automobile afin d'optimiser l'estimation et la planification des réparations.

- La réglementation et les cadres de soutien à la finance ouverte, tels que l'accès aux données financières (FiDA) en Europe, qui font suite aux initiatives d'open banking, obligeront les compagnies d'assurance à accroître leur capacité à partager des données via des API ouvertes.

Obstacles

- Bien que commercialisées comme des API ouvertes, les API externes sont en réalité ce que proposent la plupart des assureurs. Pour conserver un certain niveau de contrôle sur l'utilisation, les assureurs exigent généralement un engagement avec des partenaires avant l'utilisation.

- Il est important que les DSI reconnaissent que les API ouvertes nécessitent de connecter la stratégie commerciale, la stratégie marketing et la stratégie technologique. Les initiatives menées par l'informatique risquent d'être abandonnées.

- Par rapport à d'autres secteurs d'activité de services financiers tels que la banque, la génération de normes et l'acceptation dans le domaine de l'assurance ont été bien inférieures, ce qui entrave la mise à l'échelle.

- Bien que l'échange de valeur sans friction soit techniquement possible, il peut ne pas être souhaitable pour des raisons commerciales ou de conformité. Par exemple, le changement de produit ou de fournisseur peut souvent être un processus difficile qui nécessite le recours à des conseils professionnels.

- Les cas d'utilisation actuels se concentrent sur des parties spécifiques de la chaîne de valeur de l'assurance, telles que l'agrégation de données, les devis et les nouvelles affaires, et principalement sur des produits P&C plus simples tels que l'assurance voyage et les gadgets.

Recommandations pour les utilisateurs

- Créez un argument en faveur de l'investissement dans les API ouvertes en travaillant avec les parties prenantes de l'entreprise pour identifier les points d'intégration dans les futurs modèles commerciaux d'assurance ouverts qui impliquent des partenaires externes.

- Favorisez les relations avec des partenaires externes en identifiant des transactions communes pour aider à financer la co-création de nouvelles API ouvertes.

- Encouragez les partenaires à expérimenter la création de nouveaux services ouverts basés sur des API en créant des environnements sandbox qui nécessitent un examen et un enregistrement moins rigoureux que les versions de production afin de réduire les obstacles à l'adoption.

- Gérez la consommation d'API tierces de la même manière que vous gérez les applications SaaS. Négociez et appliquez les accords de niveau de service, la conformité en matière de sécurité, les conditions de licence et la continuité du service.

- Surveillez la disponibilité des API ouvertes et externes dans tous les secteurs susceptibles de répondre aux besoins des entreprises d'assurance en suivant les cas d'utilisation du secteur sur les marchés des API et via les fournisseurs de technologies d'assurance .

Lectures recommandées par Gartner

Pour se préparer aux opportunités d'assurance ouvertes, les DSI doivent investir stratégiquement dans les API

Étude de cas : une transformation de l'écosystème numérique pilotée par les API d'assurance

Comment concevoir des API performantes

Comment utiliser les KPI pour mesurer la valeur commerciale des API

Passerelles API fédérées

Analyse par : Shameen Pillai

Évaluation des avantages : élevée

Pénétration du marché : 1 à 5 % du public cible

Maturité : Émergente

Définition:

Les passerelles API fédérées combinent des instances de passerelle hétérogènes gérées par un plan de contrôle commun. Le plan de contrôle permet une gouvernance centralisée et prend en charge l'automatisation des modifications de stratégie API. Pour répondre à des cas d'utilisation spécifiques, les passerelles API individuelles peuvent varier en termes de modèle de déploiement, d'ensemble de fonctionnalités et d'étendue de responsabilité.

Pourquoi c'est important

Les organisations doivent pouvoir utiliser différentes technologies de passerelle API dans leurs architectures dans différentes circonstances . Cela est souvent dû à l'adoption de services de passerelle API natifs d'un fournisseur de cloud, à des fusions et acquisitions et à une sélection de passerelles disparates pour répondre aux exigences fonctionnelles ou opérationnelles dans différents domaines (par exemple, API ouvertes, API B2B ou API internes). Cependant, le défi réside dans la mise en œuvre d'une administration centrale de diverses passerelles . Mais une fois accomplies, les passerelles API fédérées permettent la gestion des passerelles API multicloud et multifournisseurs pour l'évolutivité, la cohérence, la sécurité et une expérience unifiée pour les consommateurs et les fournisseurs d'API.

Impact sur les entreprises

La fédération coordonne les passerelles API multifournisseurs . Cela permet aux entreprises de bénéficier des avantages des technologies API natives dans les environnements multicloud et sur site, tout en minimisant la complexité de la gouvernance et de l'administration. Les passerelles API fédérées peuvent aider les entreprises à se conformer aux réglementations du secteur

concernant l'emplacement des données et les normes API, à créer une architecture agile pour l'innovation et à mieux assimiler les systèmes acquis par le biais de fusions et acquisitions.

Conducteurs

- Les entreprises utilisent plusieurs passerelles API natives du cloud, ainsi que des passerelles API provenant de sources open source et commerciales.
- Les entreprises utilisent différentes passerelles API pour répondre à différents modèles d'implémentation et satisfaire à des cas d'utilisation spécifiques. Par exemple, les passerelles d'entreprise et départementales offrent des fonctionnalités étendues et grossières nécessaires pour gérer le trafic externe et servir les applications. Ces passerelles sont équipées pour traiter des volumes de trafic élevés et de grands ensembles d'API. Derrière les passerelles API d'entreprise, les développeurs préfèrent les passerelles légères pour faire face à des ensembles d'API plus petits ou des API hébergées dans des environnements différents. Celles-ci ont une faible empreinte et peuvent être déclaratives pour faciliter le déploiement et la mise à l'échelle. Les ingénieurs en fiabilité des sites trouvent les passerelles API légères convaincantes en raison de leur efficacité et de leur simplicité opérationnelle.
- Les organisations utilisant une posture de sécurité zéro confiance s'appuient sur une hiérarchie de passerelles API pour mettre en œuvre des mesures de sécurité dans le flux d'API orchestrées.
- Le paysage des passerelles API continue d'élargir ses offres, alors que les solutions de passerelle se battent pour être l'option préférée pour un ou plusieurs modèles d'architecture et cas d'utilisation d'API.

Obstacles

- Les technologies de passerelle API ne sont pas toutes à égalité avec les capacités de médiation, et il n’existe pas d’ensemble de fonctionnalités de médiation de base communément attendu.
- Il existe un manque marqué d’intérêt parmi les fournisseurs pour rendre leurs passerelles API compatibles avec les plans de contrôle d’autres fournisseurs.
- L’absence de normalisation des politiques conduit les fournisseurs à avoir des opinions indépendantes sur les politiques de passerelle et sur la manière dont elles sont définies.
- Les mécanismes et capacités de sécurité varient selon les passerelles API.

Recommandations pour les utilisateurs

- De nombreux fournisseurs de passerelles API proposent des variantes qui peuvent être déployées dans plusieurs environnements cloud et sur site. Recherchez des solutions qui optimisent les passerelles du même fournisseur dans votre architecture tout en répondant à vos exigences.

- Identifiez clairement les politiques d'API qui doivent être cohérentes dans toute l'entreprise. Idéalement, les politiques liées à la sécurité, à l'authentification, à l'utilisation et à la gestion du trafic doivent être ciblées pour être appliquées dans les passerelles d'API. La médiation de protocole et de format (SOAP vers REST, XML vers JSON) et le marshaling et l'orchestration simples des données sont également présents dans les passerelles d'API, mais évitez d'inclure des processus complexes et une logique métier.
- Prenez en charge la centralisation de la surveillance et de l'analyse des API sur des passerelles API hétérogènes en utilisant des solutions qui capturent et regroupent les journaux de passerelle et les mesures d'utilisation des API. Les solutions OpenTelemetry deviennent également de plus en plus répandues.
- À mesure que les passerelles API deviennent banalisées, il faut mettre au défi les fournisseurs de parvenir à un consensus sectoriel sur des ensembles de politiques standard pour elles .

Fournisseurs d'échantillons

APIIDA; APIwiz; Axway

Au sommet

Produits API

Analyse par : John Santoro, Nicholas Carter

Évaluation des avantages : élevée

Pénétration du marché : 20 à 50 % du public cible

Maturité : Début de la généralisation

Définition:

Un produit API est un ensemble d'API regroupées par fonctionnalité et généralement proposées dans le commerce, mais qui peuvent être proposées gratuitement ou en interne. Tous les produits API doivent impliquer un chef de produit, un support technique, une feuille de route et des conditions commerciales spécifiant l'utilisation acceptable, les limites d'utilisation et les SLA. Les produits API mis à la disposition des clients et des partenaires sont susceptibles d'être découverts via un portail API destiné à améliorer la productivité des développeurs qui les utilisent.

Pourquoi c'est important

De plus en plus, les responsables de l'ingénierie logicielle voient le potentiel commercial de proposer leurs API à des clients ou à d'autres organisations au sein de leur entreprise. En plus d'améliorer les produits existants, les API peuvent être vendues à de nouveaux publics clients ou constituer la base d'un modèle commercial de plateforme. Pour monétiser les API de cette manière, il faut les traiter comme des produits et non comme de simples composants internes à utiliser « tels quels ».

Impact sur les entreprises

Les produits API peuvent fournir une nouvelle source de revenus et améliorer la fidélisation. Les produits API créés à partir de capacités et de données commerciales permettent aux entreprises de vendre à des publics entièrement nouveaux. Lorsque les produits API permettent un modèle commercial de plateforme, les entreprises peuvent créer une solution plus complète grâce aux contributions des partenaires qui s'appuient sur la plateforme.

Conducteurs

- La monétisation des informations, des données et des processus pour les clients existants ou nouveaux nécessite un nouveau produit ou un nouveau mécanisme de livraison du produit.
- La création d'un modèle de plateforme pour les partenaires et les clients nécessite l'accès aux API qu'ils utiliseront pour créer de nouvelles solutions.
- Les clients qui souhaitent créer des applications composites ont besoin d'un catalogue de produits API.
- Les produits API offrent des moyens alternatifs de consommer les fonctionnalités des produits dans les applications composites.

Obstacles

- Les équipes techniques qui implémentent les API supposent souvent que les API sont précieuses sans avoir analysé le retour sur investissement potentiel qu'elles pourraient représenter en les proposant en tant que produit API.
- Les équipes d'ingénierie pensent que la publication d'API internes en externe suffira à faire le travail, alors que les clients peuvent avoir besoin d'API différentes de celles utilisées en interne.
- Les produits API sont facilement clonés par les concurrents, qui peuvent simplement créer un produit similaire avec les mêmes API. Cela présente des défis concurrentiels.
- Les produits API nécessitent plus que l'investissement technique nécessaire à la conception, à la mise en œuvre et à la maintenance des API. Ils nécessitent également un support opérationnel avec des SLA, un support technique pour les clients, ainsi que la création et la maintenance d'un portail API, de documentation, de tutoriels et de supports de formation.

Recommandations pour les utilisateurs

- Organisez-vous pour maximiser le succès du produit API en établissant une équipe de gestion de produit responsable de la feuille de route du produit, des ventes, du marketing et du support.
- Assurez le succès du produit API sur le marché en concevant le produit en fonction de ce que veulent les clients, et non de ce que les développeurs ont produit.
- Justifiez le coût de l'investissement dans un produit API en créant un plan d'affaires qui comprend le marché adressable, un modèle de tarification et un prix probables, ainsi qu'une comptabilité complète des coûts de l'organisation.

Fournisseurs d'échantillons

Checkr; Cortical.io; Courier ; Éligible

Lectures recommandées par Gartner

L'évolution du rôle du chef de produit API dans la gestion des produits numériques

Collaborez avec les chefs de produit pour garantir le succès des produits basés sur les API

Les chefs de produit doivent regarder au-delà des développeurs pour comprendre les modèles d'achat des API

Coup de projecteur sur la vidéo : la stratégie API de Ford Pro

Comment fixer le prix et commercialiser des API monétisées pour une adoption maximale

SaaS sans tête

Analyse par : l'équipe de recherche en génie logiciel

Évaluation des avantages : élevée

Pénétration du marché : 1 à 5 % du public cible

Maturité : Émergente

Définition:

Headless SaaS est un service d'application cloud conçu avec des interfaces de programmation (API) comme principale méthode d'accès (au lieu d'une interface utilisateur traditionnelle, désormais facultative). Ces services sont également qualifiés de centrés sur les API. Headless SaaS fournit un ensemble de composants ou de services logiciels d'entreprise qui peuvent être consommés par d'autres applications ou composés d'autres composants pour fournir des solutions.

Pourquoi c'est important

Le SaaS headless permet une flexibilité et une composabilité basées sur les fonctionnalités fournies par les fournisseurs. Il fournit des logiciels d'entreprise modulaires fournis sous forme de blocs de construction compatibles API que les clients peuvent utiliser pour étendre d'autres applications ou pour utiliser dans le développement d'applications personnalisées. Il offre aux clients plus de créativité et de flexibilité dans la création de solutions. Une plus grande créativité dans l'ingénierie des applications se traduit par une autonomisation des entreprises pour une innovation plus rapide, plus sûre et plus efficace.

Impact sur les entreprises

Les entreprises équipées pour utiliser le SaaS headless créent de nouvelles expériences applicatives pour leurs employés et leurs clients grâce à la composition de composants logiciels nouveaux et prédéfinis, parfois issus de plusieurs fournisseurs ou applications. Elles ont ainsi accès à des innovations plus percutantes pour mieux s'adapter aux besoins changeants des utilisateurs et mieux répondre aux opportunités concurrentielles.

Conducteurs

- La conception d'applications modernes repose sur l'intégration et la composition inter-applications, ce qui encourage les fournisseurs d'applications à fournir leurs fonctionnalités logicielles sous forme de services sans tête accessibles de manière facultative ou principalement via des API.
- Les entreprises exigent la possibilité de personnaliser leurs applications et de les enrichir avec des fonctionnalités de plug-ins tiers . Les fournisseurs SaaS doivent permettre à leurs clients de réorganiser leurs fonctionnalités métier .
- La conception d'applications modulaires et composables, axées sur les API, modifie les critères d'évaluation des utilisateurs pour les SaaS, en faveur de la compatibilité. Le groupe de défense MACH Alliance, dirigé par les fournisseurs, promeut l'architecture SaaS headless comme principe architectural fondamental pour le commerce numérique et d'autres solutions.
- De nombreuses applications plus anciennes sont accessibles via des API pour les inclure dans la modernisation et l'innovation de l'informatique des organisations.
- La conception des applications métier est systématiquement divisée en plusieurs expériences front-end et services back-end distribués compatibles avec les API. Cette architecture est appelée architecture d'applications et de services en maillage (MASA). Chaque partie utilise des outils et une expertise de conception différents. Un nombre croissant de fournisseurs d'applications se concentrent sur la logique métier et les données back-end et laissent l'expérience utilisateur (UX) à des équipes distinctes, y compris les propres développeurs du client.

Obstacles

- Les équipes peuvent manquer des compétences et des outils nécessaires pour créer ou utiliser des composants composables.
- Les meilleures pratiques en matière de tarification et d'approvisionnement de SaaS headless ne sont pas bien développées, ce qui retarde l'adoption ou augmente ses coûts. La tarification des API à usage occasionnel dans les SaaS traditionnels est généralement onéreuse et ne correspond pas aux pratiques d'utilisation des produits d'application axés sur les API.
- L'utilisation de composants headless multisources pour créer de nouveaux processus et expériences d'application nécessite un travail d'intégration qui peut ne pas être pris en charge dans certains outils de composition.
- Les fournisseurs peuvent proposer l'accès à leurs API en tant que citoyens de seconde classe par rapport à leur interface utilisateur préférée. Bien que les API puissent être présentes, elles peuvent ne pas être bien documentées ou ne pas avoir de feuille de route claire.
- Les interfaces utilisateur réduites ou absentes fournies avec les SaaS headless supposent et exigent que le client implémente sa propre application et son expérience utilisateur

différenciées . Ce qui constitue une opportunité d'innovation bienvenue pour certains peut être un fardeau pour d'autres, retardant l'adoption du SaaS centré sur les API.

Recommandations pour les utilisateurs

- Développez les outils et les compétences de gestion des API qui reconnaissent les exigences supplémentaires pour régir l'accès aux API tierces importées.
- Privilégiez les offres SaaS qui exposent leurs fonctionnalités métier sous forme d'API et/ou de flux d'événements.
- Planifier l'utilisation croissante de la composition et de l'intégration de logiciels d'entreprise centrés sur les API dans la conception et la fourniture de services, de processus et d'expériences d'application.
- Assurez une séparation claire basée sur l'API de la logique métier back-end et de l' interface utilisateur front-end dans la plupart des applications d'entreprise pour maximiser les avantages à long terme du SaaS headless adopté .
- Privilégiez les offres de plateformes applicatives bien équipées pour un accès géré aux API externes et aux sources d'événements.
- Pratiquer l'utilisation et la gouvernance des API en vue d'une plus grande adoption du SaaS headless.
- Soyez à l'affût des opportunités d'expérimenter un nouveau modèle commercial en proposant certaines de vos fonctionnalités commerciales sous forme de produits ou de services API payants.

Fournisseurs d'échantillons

Algolia; Logiciel Alloy ; Oiseau ; Clearbit; Cloudinary; Lob ; Plaid; Strapi; Stripe; Twili o

Lectures recommandées par Gartner

Prédictions pour 2024 : la modularité composable façonne la nouvelle base numérique

Innovation Insight pour la composition d'expériences numériques

Quand les applications doivent-elles utiliser une approche d'architecture MACH ?

Collaborez avec les chefs de produit pour garantir le succès des produits basés sur les API

Réponse rapide : ce que les GM doivent savoir sur l'avenir composable des applications

API FHIR

Analyse par : Mandi Bishop, Roger Benn, Laura Craft

Évaluation des avantages : élevée

Pénétration du marché : 5 à 20 % du public cible

Maturité : Émergente

Définition:

Les API HL7 Fast Healthcare Interoperability Resources (FHIR) représentent une norme moderne pour l'échange de données de santé entre les participants de l'écosystème tels que les payeurs, les prestataires, les entreprises des sciences de la vie, les organismes de réglementation, les prestataires de services sociaux et les patients. FHIR permet le partage de données pour des domaines tels que les déterminants administratifs, cliniques et sociaux. FHIR est open source et basé sur des normes Internet largement adoptées telles que REST et JSON.

Pourquoi c'est important

L'interopérabilité des données de santé est restée un sujet de préoccupation pendant des décennies, les normes propriétaires dominant les systèmes de base et les exigences de reporting. Les API FHIR fournissent une norme commune ouverte pour l'échange sécurisé de données entre les participants de l'écosystème de la santé tels que les prestataires, les payeurs et les organisations des sciences de la vie (collectivement appelés HCLS) ainsi que les parties prenantes de la société. FHIR fait l'objet d'un développement et d'un perfectionnement continus par la communauté mondiale HL7, ce qui le rend extensible et adaptable.

Impact sur les entreprises

API FHIR :

- Fournir une norme mondiale commune pour l’échange de données liées à la santé.
- Accélérez le développement et la mise en œuvre de solutions grâce aux normes Web, telles que REST, XML, JSON, HTTP et OAuth.
- Établir une utilisation des informations basée sur des normes (telles que des lignes directrices de mise en œuvre de la pratique clinique) qui améliore la collaboration et l’intégration des flux de travail.
- Réduisez le temps de rentabilisation des initiatives de partage de données.
- Sont désormais pris en charge par les plateformes de gestion d'API, de low-code et d'intégration.

Conducteurs

- L'intérêt et le soutien des gouvernements pour les normes de données de santé augmentent dans plusieurs régions du monde. Quatre-vingt-quatre pour cent des personnes ayant répondu à une **enquête mondiale de septembre 2023 menée par HL7** (représentant 24 pays différents) s'attendent à ce que l'adoption de FHIR augmente dans les années à venir. Le même nombre a indiqué qu'une réglementation axée sur FHIR est en place. Près de la moitié s'attendent à une forte augmentation de l'adoption de FHIR. Seuls trois pays ont indiqué qu'aucun cas d'utilisation de FHIR n'était actuellement utilisé pour échanger des données dans leur pays.

- Les accords de soins fondés sur la valeur qui s'étendent à tous les secteurs des services de santé et de santé gagnent du terrain. Ces accords nécessitent un partage étendu des données, par exemple la participation des fabricants de produits pharmaceutiques à la gestion des maladies chroniques et la prise en charge des risques liés aux résultats cliniques.

- Les API basées sur des normes ouvertes réduiront à terme la complexité et le coût de l'échange de données pour l'administration des soins de santé en limitant la traduction laborieuse des données nécessaire entre les normes propriétaires de chaque entité pour autoriser les services, examiner la documentation clinique ou payer les demandes de remboursement.

- Il existe un marché florissant de solutions de serveur et de plateforme de données FHIR qui comprend des applications open source, tous les principaux fournisseurs de services cloud et des fournisseurs de solutions de niche. En plus des serveurs FHIR, de nombreux fournisseurs proposent des solutions compatibles FHIR telles que les dossiers médicaux électroniques (DME), les moteurs de réclamation, les systèmes de gestion des soins et les plateformes d'analyse de la santé de la population.

- Les applications médicales substituables et les technologies réutilisables à haute valeur ajoutée (SMART) – collectivement appelées applications « SMART on FHIR » – sont de plus en plus disponibles dans les principaux DSE. Ces cas d'utilisation incluent l'autorisation préalable, l'identification des lacunes en matière de soins et le codage de l'ajustement des risques. Cela donne un nouvel élan au FHIR lui-même.

Obstacles

- L'innovation dans les systèmes cliniques et administratifs de base du HCLS se produit lentement, voire pas du tout. Une pression importante du marché ou de la réglementation est nécessaire pour que les fournisseurs (en particulier les fournisseurs matures avec une part de marché importante) prennent en charge les API FHIR avant qu'elles ne soient largement adoptées.

- Les organisations HCLS disposent généralement d'environnements de données tentaculaires qui nécessiteraient d'importants investissements financiers et en ressources pour adopter une nouvelle façon de coder les données de santé.

- Actuellement, les difficultés techniques limitent l'adoption de FHIR à l'échelle nationale. Les intermédiaires peuvent interrompre l'échange de données au lieu de le faciliter. La mise en correspondance des identités des patients entre les différentes parties prenantes n'est pas fiable.

- Les approches d'authentification et d'autorisation varient, menaçant la confidentialité. Il n'existe aucun répertoire de points de terminaison FHIR géré de manière centralisée.

- Les données de santé sont une cible de choix pour les pirates informatiques. La sécurité des API varie d'un système à l'autre. Les DSE sont généralement des environnements sécurisés, tandis que les applications mobiles peuvent stocker des clés et des jetons d'API en texte clair.

- Les normes FHIR évoluent constamment, créant une cible mouvante pour la mise en œuvre et la maintenance.

Notes d'analyste : En raison du grand intérêt suscité, d'un marché de fournisseurs en pleine croissance et en pleine maturité, et des obligations réglementaires qui poussent à l'adoption de l'API FHIR, cette innovation est au sommet des attentes. Cependant, des obstacles importants empêchent d'atteindre un retour sur investissement, il faudra donc jusqu'à 10 ans pour parvenir à une adoption généralisée à l'échelle mondiale. L'adoption sera beaucoup plus rapide dans les régions où les exigences réglementaires sont strictes. Les fournisseurs représentatifs ont une présence mondiale et proposent des services FHIR, car FHIR lui-même est open source.

Recommandations pour les utilisateurs

- Participez aux groupes de travail FHIR qui définissent des modèles et des profils logiques spécifiques aux cas d'utilisation et relèvent les défis techniques liés à la mise à l'échelle. Il s'agit notamment de :
  - Le projet Da Vinci (pour les cas d'utilisation administrative)
  - Le Projet Gravity (pour les déterminants sociaux de la santé)
  - Projet d'harmonie des genres (pour l'orientation sexuelle et l'identité de genre)
  - Projet Vulcan (pour la recherche clinique et translationnelle)
  - Projet X-eHealth (pour l'interopérabilité transfrontalière de l'UE)
  - Hélios (pour la santé publique)
  - Le groupe de travail FHIR à grande échelle (FAST) (pour résoudre les défis techniques)
- Évaluez les fournisseurs de systèmes de base et de plateformes de données existants pour leurs capacités de support d'API FHIR et leurs plans de feuille de route.
- Intégrez les API FHIR à votre stratégie de gestion des API.
- Collaborez avec les partenaires HCLS régionaux pour identifier les opportunités d'utilisation de l'API FHIR à forte valeur ajoutée. Pilotez (ou développez, si elles existent) les connexions pour établir les coûts de base, votre calendrier de mise en œuvre et vos objectifs de performance initiaux.

Innovation dans la pratique : en juillet 2023, l'Organisation mondiale de la santé (OMS) et HL7 ont convenu de soutenir les classifications et terminologies internationales dans FHIR et de fournir des spécifications de normes et un environnement de test. Les mandats américains exigent FHIR pour un certain nombre de cas d'utilisation. Le système national de santé (NHS) au Royaume-Uni utilise FHIR. Le projet X-eHealth développe un échange de données basé sur FHIR à l'échelle de

l'UE. Le réseau national de données sur la santé du ministère brésilien de la Santé utilise FHIR. Israël a proposé une législation pour rendre FHIR obligatoire.

Fournisseurs d'échantillons

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

Lectures recommandées par Gartner

Comment l'intégration des données cliniques améliore l'interopérabilité des systèmes de paiement des soins de santé aux États-Unis

Références et tendances en matière d'interopérabilité des prestataires de soins de santé, 3e trimestre 2023

Références en matière d'interopérabilité des organismes payeurs de soins de santé aux États-Unis, 2e trimestre 2023

API de monétisation du réseau

Analyse par : Gaspar Valdivia, Ajit Patankar, Enrique Hernandez-Valencia

Évaluation des avantages : Transformationnel

Pénétration du marché : 1 à 5 % du public cible

Maturité : Émergente

Définition:

Les API de monétisation de réseau font référence aux interfaces de programmation d'applications que les développeurs de logiciels et les partenaires commerciaux peuvent utiliser pour accéder par programmation afin de récupérer directement des données, de demander des services ou de modifier les capacités de service des réseaux de télécommunications. Ces API sont conçues pour faciliter l'intégration, permettant la monétisation des actifs du réseau et prenant en charge une variété de modèles commerciaux qui peuvent aller des frais de transaction aux abonnements et au partage des revenus tout au long de la chaîne de valeur.

Pourquoi c'est important

Les API de monétisation des réseaux créent les bases de nouvelles façons pour les fournisseurs de services de communication (CSP) de rentabiliser les investissements importants qu'ils continuent de faire dans les réseaux pour la couverture, la capacité et la mise en place de nouveaux services. La GSM Association (GSMA) sponsorise l'initiative Open Gateway en partenariat avec le projet CAMARA de la Linux Foundation pour utiliser les outils et les meilleures pratiques de la communauté mondiale des développeurs. L'objectif est de fournir des API de monétisation des réseaux de manière contrôlée, homogène et fédérée.

Impact sur les entreprises

Les API de monétisation de réseau peuvent créer une nouvelle économie d'API autour des réseaux de télécommunications . Elles peuvent prendre en charge de nouveaux cas d'utilisation pour les

entreprises et les organismes publics ou améliorer l'expérience utilisateur dans d'autres cas existants en permettant l'utilisation des données récupérées à partir du réseau et des capacités du réseau de nouvelles manières. Les exemples incluent la détection et la prévention des fraudes, l'optimisation des coûts d'itinérance, la facturation des opérateurs et la qualité du réseau à la demande pour une variété de cas d'utilisation tels que les appareils de réalité augmentée/réalité virtuelle (AR/VR) ou les véhicules autonomes.

Conducteurs

- Le projet CAMARA fournit un cadre permettant aux opérateurs de réseau et aux partenaires de concevoir, de mettre en œuvre et de gérer le développement d'un ensemble commun d'API de services nord-nord cohérentes et interopérables pour exposer les capacités du réseau.

- Depuis le lancement d'Open Gateway en février 2023, plus de 239 opérateurs de réseau ont rejoint l'initiative et plus de 40 ont lancé commercialement au moins une de leurs API de monétisation de réseau. Parmi ces API, citons la vérification du numéro, la connaissance du client/ MATCH , la localisation de l'appareil, l'échange de cartes SIM et la facturation par l'opérateur.

- Les fournisseurs d'équipements réseau tels qu'Ericsson (via leur unité Vonage) et Nokia (via leur approche Network as Code) soutiennent activement le développement d'un écosystème d'API de monétisation de réseau. Ils investissent dans des plateformes d'agrégation d'API, créent des cadres commerciaux et des canaux de distribution pour les API réseau sur l'ensemble des réseaux et des pays, et fournissent des outils pour aider les CSP à lancer des API réseau et les développeurs à les utiliser.

- Les hyperscalers cloud tels qu'Amazon Web Services et Microsoft Azure participent également au projet Open Gateway et ont commencé à distribuer les API de monétisation du réseau des CSP via leurs places de marché pour les développeurs de logiciels, servant de canal puissant pour aider les développeurs à découvrir et à utiliser ces API de réseau de télécommunications.

- Les modèles d'API réseau composables qui ont déjà fait leurs preuves auprès des fournisseurs de plateformes de communication en tant que service (CPaaS) tels que Twilio, Infobip et Sinch sont également intégrés aux nouvelles spécifications d'API réseau. Les fournisseurs de CPaaS se sont concentrés sur les services de télécommunications de base tels que les SMS et les liaisons SIP et peuvent désormais être étendus à la communauté 5G au sens large .

- La possibilité d'exposer certaines API standard à l'usage des entreprises et des développeurs pour répondre aux besoins des réseaux mobiles privés favorise encore davantage cette innovation.

Obstacles

- Le développement de l'économie des API de monétisation des réseaux nécessite un soutien au-delà d'un CSP isolé. Pour attirer les développeurs, un ensemble suffisamment large de CSP dans un pays ou une zone géographique donnée doit proposer systématiquement les API réseau nécessaires à la mise en œuvre de cas d'utilisation de réseaux inter-opérateurs.

- Les développeurs souhaitant adopter des API réseau s'attendront à ce qu'elles fonctionnent sur plusieurs CSP. Si les CSP ne parviennent pas à prendre en charge les fonctionnalités inter-réseaux (comme cela s'est produit précédemment avec la messagerie des services de communication enrichis [RCS]), toute la proposition de valeur est menacée.

- Des modèles commerciaux et d'affaires, y compris des mécanismes de règlement, qui satisfont tous les acteurs de la chaîne de valeur — fournisseurs de réseau, agrégateurs d'API, hyperscalers, potentiellement intégrateurs de systèmes, développeurs de logiciels, entreprises et utilisateurs finaux — sont nécessaires.

- La nécessité d'adapter les modèles commerciaux, les modèles commerciaux et les niveaux de prix pour la monétisation des API aux cas d'utilisation et aux zones géographiques peut donner lieu à de longues négociations, retarder la mise sur le marché en tant que capacité mondiale et diminuer l'attrait pour les développeurs de logiciels s'ils ne sont pas suffisamment automatisés.

- Certaines API de monétisation de réseau, telles que « Quality on Demand » , nécessitent le déploiement de technologies plus complexes et des investissements réseau supplémentaires. Elles peuvent également nécessiter des changements de réglementation sur certains marchés. Ces obstacles se traduiront par une disponibilité inégale selon les opérateurs et les zones géographiques pendant une longue période.

Recommandations pour les utilisateurs

- Collaborer avec d'autres CSP via la GSMA Open Gateway et le projet CAMARA pour assurer un développement cohérent et une mise en œuvre interopérable des API réseau avec d'autres partenaires CSP, maximiser le marché adressable et favoriser l'adoption généralisée des API de monétisation réseau par la communauté de développement de logiciels.

- Travailler en étroite collaboration avec les agrégateurs d'API, les hyperscalers et les opérateurs de marketplace d'API pour tirer parti de leurs plateformes commerciales et de leurs cadres commerciaux. Ces partenariats faciliteront la distribution, la commercialisation et la monétisation des API réseau auprès d'une communauté de développeurs plus large.

- Offrez un support complet et des ressources aux développeurs utilisant les API de monétisation du réseau. Cela comprend la documentation destinée aux développeurs, les kits de développement logiciel, les sandbox pour les tests et l'expérimentation et les canaux d'assistance dédiés.

- Communiquez de manière proactive et faites connaître les capacités et les nouvelles opportunités commerciales offertes par les API de monétisation du réseau pour accélérer l'adoption du marché et étendre la portée du marché.

- Participez aux programmes d'API ouverts de l'industrie qui offrent aux CSP des moyens de monétiser les actifs du réseau avec des modèles commerciaux basés sur l'écosystème ou la plate-forme.

API pilotées par événements

Analyse par : Max van den Berk

Évaluation des avantages : Modérée

Pénétration du marché : 5 à 20 % du public cible

Maturité : Adolescente

Définition:

Les API pilotées par événements décrivent les interfaces utilisées dans l'architecture pilotée par événements (EDA), notamment les abonnements et la création de canaux. Les API pilotées par événements diffèrent des API REST traditionnelles de type requête/réponse car elles permettent une communication asynchrone. AsyncAPI est une norme populaire pour les API pilotées par événements.

Pourquoi c'est important

Les API pilotées par événements permettent de notifier en temps réel les modifications apportées aux données et à l'état des applications. Elles constituent un moyen efficace de propager des informations à plusieurs parties et de synchroniser les modifications de données sur plusieurs applications et magasins de données. Elles prennent en charge des modèles d'intégration flexibles via des communications asynchrones, indirectes et négociées.

Impact sur les entreprises

Les API pilotées par événements ouvrent des opportunités commerciales pour une réponse plus rapide aux changements et des analyses en continu.

Elles facilitent et aident également à normaliser l'EDA . Les API pilotées par événements offrent des avantages tels que l'activation des notifications push, qui sont beaucoup plus efficaces et moins coûteuses - tant du point de vue du temps que du réseau - que les sondages.

Conducteurs

- L'analyse des données et l'IA génèrent de nouveaux cas d'utilisation liés aux événements et aux notifications, qui à leur tour entraînent le besoin d'API basées sur les événements.
- L' adoption des principes de conception d'API en premier dans l'espace API REST a généralisé les habitudes de conception de contrats en premier , augmentant ainsi leur popularité dans les applications pilotées par événements.
- La puissance de Data Fabric dépend de sa capacité à suivre les événements en temps réel, et pas seulement les magasins de données d'état opérationnel .
- Les principaux cas d'utilisation des communications basées sur les événements incluent l'agrégation de données à des fins d'analyse et d'IA, et l'accélération de la prise de décision grâce à des notifications en temps réel . Ils incluent également des scénarios d'intégration asynchrone prenant en charge les interactions un-à-plusieurs et plusieurs-à-plusieurs, ainsi que des notifications directes aux applications clientes à l'aide d'un modèle plus efficace que l'interrogation.

- Les technologies de courtage d'événements open source et cloud, en particulier Apache Kafka, ont permis de mieux faire connaître l'infrastructure de publication-abonnement, ainsi que son accessibilité, et ont encouragé l'adoption de l'EDA .

- AsyncAPI, une initiative qui définit une norme sur la manière dont les API pilotées par événements sont publiées, est soutenue par la Linux Foundation, qui est le même organisme de normalisation qui héberge la spécification OpenAPI (OAS).

- La version 3 de l'OAS, introduite en 2017 et désormais largement adoptée comme norme pour la publication d'API, inclut une disposition pour les rappels comme moyen de décrire les API pilotées par événements. La version 3.1 de l'OAS (février 2021) ajoute la prise en charge des webhooks , l'approche API pilotée par événements la plus courante pour les API orientées Internet. Il convient toutefois de noter que les webhooks sont un modèle un à un, contrairement à EDA, qui prend également en charge un modèle plusieurs à plusieurs .

Obstacles

- Les produits de médiation d'API, tels que les passerelles d'API, sont souvent créés ou configurés uniquement pour les API de requête/réponse. Dans certains cas, les modèles d'API pilotées par événements, notamment les méthodes de type « fire-and-forget » ou « publish-and-subscribe », ne sont pas pris en charge dans le produit de médiation d'API.

- La prise en charge des protocoles pour les API pilotées par événements est diversifiée, notamment les webhooks (HTTP/HTTPS), les WebSockets, Message Queuing Telemetry Transport (MQTT), Advanced Message Queuing Protocol (AMQP) et Apache Kafka. Cela rend la prise de décision plus difficile pour les responsables de l'ingénierie logicielle.

- Bien que les portails API soient largement utilisés pour la publication d'API de demande/réponse, leur utilisation pour les API pilotées par événements est naissante.

- La réutilisation des définitions de schéma, telles que Protocol Buffers (Protobuf), Apache Avro et GraphQL , n'est pas standardisée et peut nécessiter des outils de prise en charge de la part de l'utilisateur final, tels que **CloudEvents** .

- Les API pilotées par événements, comme toutes les EDA, présentent des défis en matière de débogage et de test.

Recommandations pour les utilisateurs

- Évaluez si les produits de médiation API que vous avez choisis , y compris les passerelles API dans le cadre de la gestion des API, prennent en charge les API pilotées par événements dans le cadre de votre processus de sélection technologique.

- Adoptez une approche de communauté de pratique pour améliorer les compétences globales de votre organisation en matière d'API pilotées par événements et leur relation avec l'EDA et le traitement et l'analyse des flux . Il peut s'agir d'une communauté de pratique EDA spécifique ou d'une partie d'une communauté de pratique API.

- Identifiez les opportunités d'augmenter les API de demande/réponse avec des API pilotées par événements au sein de votre portefeuille d'API pour permettre des scénarios d'intégration internes et externes.

Fournisseurs d'échantillons

Ably; Axual; Axway ; Cloud Software Group (TIBCO); Confluent; Gravitee.io; Postman; SmartBear ; Software AG; Solace

Lectures recommandées par Gartner

Réponse rapide : dois-je utiliser des outils de gestion d'API pour gérer les événements ?

Maturity Model For Event Driven Architecture

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

GenAI-Driven API Generation

Analysis By: Max van den Berk

Benefit Rating: Moderate

Market Penetration: 1% to 5% of target audience

Maturity: Adolescent

Definition:

API generation assists developers in automatically creating APIs based on code or specifications. API generation features are usually included in stand-alone API creation tools and low-code platforms, and recently in generative AI (GenAI)-powered tools. Benefits of API generation include decreased development time, consistency and security.

Why This Is Important

Gartner sees GenAI as fundamentally changing software delivery. GenAI assistance with API generation is becoming an expected capability in the software development process as it helps developers speed up their delivery (see **2024 Planning Guide for Software Development**).

Business Impact

APIs are a well-understood and common way of integrating between applications. The options for monetization that APIs offer are an important part of their business appeal. API generation offers faster design and delivery with possibilities of better consistency and governance. Gartner predicts AI coding assistants will drive developer productivity up 36% by 2028. GenAI can further speed up API generation (see **How to Communicate the Value of AI Code Assistants**).

Drivers

- The increase in pace of delivery heightens the demands on integration teams that develop and support integrations between applications.

- The increased availability of cloud-based services and API-centric SaaS in part drives a need for wider support of API types.
- Software development tools, integrated developer environments and platforms as a service all support some API generation capabilities that are accelerated by the use of GenAI.

Obstacles

- API generation has limits of extrapolation. “Garbage in, garbage out” remains true and its effects may be amplified by the addition of GenAI.
- Stakeholder involvement in the usability, correctness and effectiveness of APIs is not sped up by generative processes, but it remains vital in creating APIs that have actual demand. Skipping stakeholder involvement risks increasing unnecessary churn on APIs.
- Large language models (LLMs) are still striving to prove their reliability. Lack of confidence in GenAI-generated code can result in extensive validation processes. These time-consuming efforts offset productivity gains.
- Approaches such as retrieval-augmented generation (RAG) are needed to infuse an organization’s API inventory and API governance into the GenAI engagement. Enterprises must select products that support tactics such as including API standards and guidelines in prompts or as part of LLM training data.

User Recommendations

- Ensure guardrails for API generation by making generated APIs part of life cycle management processes that enable governance.
- Limit approaches within the organization that circumvent efforts to shift left in the development process.
- Consider other available options for generating APIs and the producer or consumer code. While GenAI is driving interest, existing options such as API management, low-code platforms or integration platforms may be more dependable.
- Choose a solution (such as RAG) to make the organization’s API inventory, standards, examples and documentation accessible to the selected GenAI technology. This enables GenAI responses (suggested APIs, code to invoke APIs, etc.) to reflect the enterprise’s APIs and comply with its API governance.
- Implement strong API security and protection that monitors all API requests and responses, including API requests from GenAI-created code for noncompliant and suspicious API activity.

Financial Data APIs

Analysis By: Don Free, Mark O'Neill

Benefit Rating: Moderate

Market Penetration: 5% to 20% of target audience

Maturity: Adolescent

Definition:

Financial data APIs provide access to financial data. They include APIs provided by banking API aggregators, which typically include balance verification and funds availability services. Customers of financial APIs include fintechs, lenders, credit reference agencies 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. Financial data APIs provide this capability using banking APIs or, decreasingly, 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 a single API 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 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.

User Recommendations

- Drive your IT organization to identify and inventory the differing financial data APIs required to support various initiatives, such as open banking, banking as a service and embedded finance.
- Reinforce the bank's oversight of banking API aggregators on their security and privacy controls.
- Establish or assign a role (e.g., chief of information flows) to monitor and rationalize the use of API provider offerings since many are expanding their transaction services.

Sample Vendors

Brankas; Envestnet | Yodlee; Fabrick; Mastercard (Finicity); MX Technologies; Plaid; Salt Edge; Token.io; TrueLayer; Visa (Tink)

Gartner Recommended Reading

Prioritize Use Cases With Gartner's Open Banking Business Priority Framework

Emerging Use Cases That Validate the Business Value of Open Banking

Modular Commerce

Analysis By: Aditya Vasudevan

Benefit Rating: Transformational

Market Penetration: 5% to 20% of target audience

Maturity: Early mainstream

Definition:

Modular commerce takes API-based (headless) commerce further by breaking down the functional modules in the core commerce platform to improve flexibility and agility. In this approach, functional modules are often separate packaged business capabilities that are mostly within the same vendor platform, have their own APIs and data model, and can be independently deployed and scaled.

Why This Is Important

The architecture for many digital commerce platforms is changing from its original monolithic form into a more componentized and API-oriented form. Modular commerce deconstructs a typical core commerce platform to increase agility and decrease interdependencies during

deployment processes. The granularity of the modules is defined by business needs, and must balance development flexibility and agility with governance, complexity and costs.

Business Impact

A modular commerce approach will provide:

- Ultimate flexibility across various customer journey touchpoints
- Product-driven experiences
- Business agility and innovation
- Diverse experiences led by business units
- Speed to market for new features
- Lower development costs as new features can be added more quickly
- Business and IT alignment to help achieve common goals

Drivers

- A modular architecture is the next step beyond API-based (headless) commerce on the path to composability, as this provides flexibility for organizations to easily replace or upgrade individual components rather than replatforming.
- Organizations that have adopted or are in the process of adopting an API-based digital commerce approach could consider a product-driven solution rather than a platform-driven one. This will enable them to innovate and scale crucial capabilities independently without impacting other capabilities.
- Modular commerce improves organizational agility by “productizing” capabilities for organizations focused on innovation and growth that want to avoid being bound by platform constraints. For example, by speeding up the innovation cycle, it can help business users keen to enhance specific experiences and capabilities.
- Modular commerce reduces dependencies across a commerce platform as the architecture enables each capability to be delivered, maintained, upgraded or replaced separately.
- Increasing number of digital commerce platforms are offering modular pricing models, giving organizations options to implement functionalities that they will use.

Obstacles

- Data dependencies exist across these capabilities, since they are mostly served by the same platform. A single-vendor modular approach tends to be a walled-garden approach, which makes it harder to replace modules with third-party alternatives.

- A more complex architecture requires a high level of digital maturity (but less than for a composable solution) from the business and entails longer initial implementation times.
- Insufficient availability may occur for specific technical skills (such as in API integration and DevOps) or business skills (such as in digital product management).
- Maintenance will be required for multiple applications supporting these modules. This involves heavy dependence on mature DevOps processes, which are needed to make this approach truly agile.

User Recommendations

- Establish, implement and stabilize a headless roadmap for your digital commerce solution; this is the key first step toward modularity.
- Ensure that, despite the headless architecture, the tools used maintain business user control over the UI and the customer journey.
- Educate senior management about the benefits of modular architecture and obtain their commitment to move to such an architecture.
- Select commerce platforms that use a modular architecture and/or review the roadmap of your existing commerce platform to help ensure that essential modules are deconstructed in an acceptable time frame.
- Establish DevOps and agile development practices to ensure timely delivery against the roadmap.

Sample Vendors

BigCommerce; commercetools; Elastic Path; Fabric; Kibo Commerce; Spryker Systems; VTEX

Gartner Recommended Reading

Choose the Right Digital Commerce Platform Architecture

What Are the Differences Between Modular and Composable Commerce Architectures?

API Observability

Analysis By: Dave Micko

Benefit Rating: High

Market Penetration: 5% to 20% of target audience

Maturity: Adolescent

Definition:

API providers are looking to get increased visibility into how their APIs operate and change, and specialist API observability solutions have emerged. Observations based on telemetry emitted from logs, performance statistics and external monitoring may be aggregated and normalized via

OpenTelemetry (OTel) standards to provide running snapshots of API performance and reliability and provide visibility of API dependencies, especially to anticipate and avoid breaking changes.

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. Developers, API platform teams and platform engineering teams are affected by API observability systems and practices.

Business Impact

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

Drivers

- API observability is key for successful adoption of 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.
- Changes to APIs impact on applications which consume those APIs, often negatively. It is therefore important to provide drift detection for APIs. This drift detection can include changes to the API definition itself, changes to code behind the API, or nonfunctional changes such as changes to the authentication scheme for the API. One of the drivers for API observability is to detect these changes.
- As APIs proliferate, understanding the dependencies among the systems consuming the APIs is critical to all phases of system design and production, 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 (see **OpenTelemetry**). 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, often impacting partners and customers. 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 the early detection of bugs, performance issues and user experience problems during the development life cycle.

Obstacles

- Although the techniques and practices of API observability can range from simple monitoring to sophisticated dependency tracking and 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. Include automation as much as possible.

- Weigh the costs associated with the benefits carefully. 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.

Sample Vendors

Atlassian (Optic); Moesif; Postman; Treblle; Tyk

Gartner Recommended Reading

Magic Quadrant for Observability Platforms

Key Considerations When Choosing APM and Observability Platforms

How to Start and Scale Your Platform Engineering Team

eBPF

Analysis By: Simon Richard

Benefit Rating: Moderate

Market Penetration: 5% to 20% of target audience

Maturity: Adolescent

Definition:

Extended Berkeley Packet Filter (eBPF) is an enhancement to the Linux operating system kernel that allows specific instruction sets to run (sandboxed) inside the kernel. It enables companies to add features to Linux without changing kernel source code or requiring kernel modules.

Why This Is Important

eBPF increases the extensibility of Linux. It allows users to create hooks that are triggered by Linux kernel events. This offers a safer and simpler way to add capabilities, such as performance, security and visibility, in Linux. Technology vendors use eBPF to avoid kernel-level modules, which carry inherent risks. eBPF is used in production at scale by hyperscalers (including Amazon Web Services, Meta and Netflix), content delivery networks (such as Cloudflare), and security vendors (such as Sysdig).

Business Impact

eBPF improves observability, security and performance for applications. However, most enterprises will not use eBPF directly. Technology vendors do use eBPF as an underpinning technology in their products and services to improve the performance and safety of programs that run on Linux. eBPF allows extremely technically savvy organizations to safely and quickly make changes to Linux. It is an improvement over alternative approaches, such as using Linux kernel modules or upstreaming to the Linux distribution.

Drivers

- Hyperscalers use eBPF to deliver more efficient cloud offerings. Networking, monitoring and security vendors, including the largest cloud-native protection platform (CNAPP) vendors, also use it in their products.
- Hyperscalers use eBPF to remediate kernel vulnerabilities without patching, to address Day 0 vulnerabilities and to more efficiently handle distributed denial of service (DDoS) attacks.
- Cisco acquired Isovalent and announced a new distributed firewall leveraging eBPF, increasing enterprise visibility and awareness of the technology.

- Organizations are looking to expedite development of software that runs on Linux by avoiding the requirement for upstream inclusion in the Linux distribution.
- Organizations are looking to improve the performance, security and monitoring capabilities of software running on Linux.
- eBPF is popular among technologically advanced companies, including technology vendors and hyperscalers, because it provides a standardized interface, supports kernel portability and requires less in-depth kernel programming knowledge.
- eBPF helps overcome the scale and visibility limitations of *iptables*, which is the default networking stack in Linux. eBPF helps optimize and customize Linux network packet handling by processing packets earlier in the cycle.
- Vendors are increasingly using eBPF in their carrier network infrastructure (CNI) software to improve performance, security and network visibility.

Obstacles

- While eBPF is realistic for technology vendors and hyperscalers, most enterprises lack the expertise and skills necessary to build and integrate eBPF-based functions.
- Most enterprises do not have the awareness, need or risk tolerance to tackle Linux kernel challenges directly.
- Many older Linux kernels don't support eBPF, or only partially support the latest features.
- Concerns about security and system reliability will severely limit what organizations are willing to deploy using eBPF, as poorly written eBPF programs can directly impact the operation of the Linux kernel.
- Integration and backward compatibility with existing non-eBPF-enabled products pose challenges.

User Recommendations

- Migrate to more modern platforms if your organization is still using Linux distributions with limited or no eBPF support.
- Seek eBBF-based Kubernetes CNI solutions when scale, performance, visibility and security are top priorities.
- Use Linux variants that provide eBPF support to enable network performance, visibility and security products.
- Configure eBPBF securely so that it does not become an attack path or vulnerability.
- Explore whether eBPF can meaningfully address your organization's performance, security or visibility challenges by supporting technologically advanced enterprises.

- Invest in eBPF, if you are a networking or network security vendor, to improve performance, enhance visibility and remain competitive.

Sample Vendors

Aqua Security; Cloudflare; Cisco; Fastly; Gigamon; New Relic; Sysdig; Tigera; Wiz

Gartner Recommended Reading

Using Emerging Service Connectivity Technology to Optimize Microservice Application Networking

Reference Architecture Brief: Software Observability

How to Protect Your Cloud-Native Applications in Production

API Monitoring

Analysis By: Nicholas Carter

Benefit Rating: High

Market Penetration: 5% to 20% of target audience

Maturity: Adolescent

Definition:

API monitoring is the practice of assessing APIs' availability and performance. It tracks, in real time, API consumers' experience, and can trigger alerts when intervention is needed. API monitoring tools typically provide security by using API traffic analysis and API usage insights to detect suspicious patterns. API monitoring is a quality assurance measure when 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 proactively resolve events. In addition, API analytics provide visibility into API usage patterns to reveal opportunities for improvement.

Business Impact

- Business outcomes depend on systems with healthy APIs; API monitoring identifies anomalies and creates alerts when APIs fail.
- 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 SLAs, regulations and other runtime obligations.
- API analytics give API product managers visibility to measure the value of APIs.

Drivers

- 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 to 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, robotic process automation (RPA) and composable platforms, API analytics provide a look into workflows and user behavior. API analysis is a valuable practice for generating insights into process improvement opportunities and new product innovations.
- APIs often invoke API orchestration. 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, website or application users of the API will be unable to log in. If a payments API becomes unavailable, an organization dependent on it will be unable to take payments. The need for API monitoring includes 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 need to monitor APIs they did not build.

Obstacles

- Architectures with diverse API technologies are both obstacles and drivers for API monitoring. Aggregating usage data across enterprise APIs can present challenges for centralized monitoring. Multiple API gateways create more urgency for unified API monitoring and visibility.
- Not all monitoring tools offer machine learning (ML)-based analytics. Some lack capabilities such as anomaly detection. This shortfall prevents valuable insight into API usage patterns, consumer behavior and performance analysis and related trends.
- Different API personas have varied requirements for monitoring that are difficult to achieve with a single tool. Operations teams need to monitor all APIs centrally to gauge system health and troubleshoot issues. However, API product managers need API monitoring tools with functionality that filters and presents select APIs as they manage their API products.
- Cloud-based API monitoring products are not generally well-suited for monitoring internal APIs and may require internal agents and the requisite developer time and understanding.

User Recommendations

- Organizations with heterogeneous API deployments must evaluate API monitoring technologies capable of aggregating across the complete landscape.

- 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 Open Telemetry Collector to take advantage of the tools adopting the emerging OpenTelemetry standard. (See **OpenTelemetry Collector**.)
- Use API monitoring tools that include ML capabilities to give your organization in-depth and advanced analytics.
- Confirm that the alerting capabilities of the API monitoring tool satisfies your requirements.
- Conduct stress tests of your API monitoring solution to confirm that the burden placed on API gateway(s) when harvesting data does not put API SLAs at risk.
- Use API monitoring tools that can take advantage of synthetic data to simulate load testing and preproduction testing.

Sample Vendors

APIContext; Datadog; Kong; Moesif; New Relic; SmartBear; Tyk

Sliding into the Trough

API Management for Healthcare

Analysis By: Roger Benn, Gregg Pessin

Benefit Rating: High

Market Penetration: 5% to 20% of target audience

Maturity: Early mainstream

Definition:

API management for healthcare includes IT tools and platforms for creating, provisioning, monitoring and maintaining APIs. Comprehensive API management consists of the entire API life cycle. The increased adoption of Health Level Seven (HL7) Fast Healthcare Interoperability Resources (FHIR) and the emergence of interoperable application ecosystems have made API management an increasingly important IT capability indicator of real-time health system maturity.

Why This Is Important

Mobile apps, modern web architectures and the Internet of Things (IoT) have positioned APIs as essential interoperability components of any digital strategy. APIs are central to healthcare's digital transformation and the centerpiece of the healthcare industry's evolving approach to interoperability. Successful APIs will have many active consumers and must be secured, monitored, maintained and managed throughout their life cycle.

Business Impact

Appropriate API management drives healthcare governance, integrity and performance:

- APIs allow providers to access patient data from various sources, streamline care coordination and enhance clinical decision making.
- APIs ensure efficient exchange of data with payers, pharmaceutical, medical device and patient-facing technology supporting provider access to real-world data, integration with electronic health record (EHR) and remote patient monitoring to aid in personalized care.

Drivers

- While APIs and service-oriented principles have been around for some time, they continue to gain acceptance and traction by vendors in the healthcare provider market. Healthcare delivery organizations (HDOs) require more timely responses to their business and digital transformation requirements than the healthcare software vendor community can accommodate through release cycles and product roadmaps.
- Before FHIR, healthcare APIs were often proprietary, and data sharing between different healthcare systems was challenging due to the lack of a standard format for exchanging healthcare data. FHIR introduced a universal standard for healthcare data exchange, enabling healthcare systems to communicate and share data more efficiently and effectively.
- The adoption of API management continues to increase, as the pace of healthcare provider business and clinical information-sharing requirements increases, along with strategic digital transformation initiatives facilitated by interoperability advances. The introduction of composable application architecture is accelerating interest in API management.
- In the U.S., patient access and interoperability requirements codified by the Office of the National Coordinator and Centers for Medicare & Medicaid Services interoperability rules drive API management adoption. These rules require open APIs for healthcare data access and exchange.
- HDOs expect proprietary and open APIs from their vendor community that can be safely consumed and orchestrated to support new data requirements, workflows and business capabilities.

Obstacles

- Vendors lacking healthcare experience are marketing API management solutions to the healthcare industry. This may result in solutions that will not easily accommodate traditional healthcare workflows, use cases and information exchange patterns.
- Some solutions are only available as cloud-only, which may be a limiting factor for healthcare, as some organizations have a lingering preference for on-premises deployment.
- The pricing and subscription models of the various API management platform vendors can be at odds with the high data transaction volumes of typical healthcare provider integration and data exchange workflows. Developing, maintaining and securing healthcare APIs require ongoing investment.

User Recommendations

- Implement an API management program to streamline the delivery of new business capabilities, extend existing applications and systems such as the EHR, and enable mobile and other multichannel clients.
- Leverage FHIR API opportunities within the data integration ecosystems to efficiently expose data and functionality through API-managed protocol.
- Utilize API management technologies to help build, consume, operate, secure and manage self-developed APIs and FHIR resources. Use API management platforms to centralize authentication and authorization for the APIs.
- Source your API management capabilities from purpose-built API management, clinical data interchange platforms and the existing interfacing/integration platform.
- Employ APIs when conventional industry interoperability messaging standards fall short of the health information and workflow needs.

Sample Vendors

Axway; Boomi; Cloud Software Group (TIBCO); Google (Apigee); IBM; Kong; Microsoft; Salesforce (MuleSoft); WSO2

Gartner Recommended Reading

Comparing Architectures for Hybrid and Multicloud API Management

How to Evaluate API Management Solutions

Magic Quadrant for API Management

Reference Model for API Management Solutions

MACH

Analysis By: Anne Thomas

Benefit Rating: Low

Market Penetration: 1% to 5% of target audience

Maturity: Emerging

Definition:

MACH is a useful umbrella term to describe modern architectural principles that facilitate the creation of composable components and API-centric SaaS products. MACH is an acronym of the four MACH principles: microservices, API-first, cloud-native SaaS, and headless. The term was coined by the MACH Alliance, which is a vendor-driven marketing effort that supports composability in digital commerce and other related digital experiences.

Why This Is Important

MACH originated from vendors, in particular the vendor commercetools, and has a strong vendor-driven community via its advocacy group MACH Alliance. MACH defines a set of architecture principles for creating modular application components. Its initial focus has been to enable composability in e-commerce solutions, although its modularity principles are relevant to other use cases and can be applied to any service to create a composable component.

Business Impact

MACH principles facilitate composability. MACH Alliance certification indicates that a product may be relatively easy to interconnect with other products or applications in API-centric marketplaces. MACH enables small and startup vendors to enter markets dominated by large vendors by offering easily interconnected add-on capabilities. Vendors can use MACH to make it easier for customers and partners to identify products that meet their criteria.

Drivers

Composability: All four MACH principles support composability:

- Microservices ensure that the functionality is encapsulated and cohesive. (The MACH definition of “microservice” is from a business perspective, as in API-centric SaaS components, rather than a technical architecture perspective.)
- API-first ensures that all functionality in the service is accessible via an API.
- Cloud-native SaaS ensures that the software is easy to consume, offered as a service, managed by the vendor, and supports multitenancy.
- Headless ensures that consumers can use the functionality as they wish, and they aren’t forced to use a prescribed UI to access the functionality. The Alliance includes vendors of UI composition tools.

Vendor marketing: The MACH Alliance is a vendor-driven initiative to facilitate markets for composable componentry.

- MACH Alliance certification and inclusion in the MACH Alliance members directory provide visibility and cachet to vendors that offer API-centric SaaS componentry.
- Small and startup vendors can design componentry that works within larger application systems, offering them an opportunity to break into new markets with innovative capabilities.
- Large vendors benefit when small vendors deliver new features and capabilities that complement and extend their core applications.

Obstacles

- While the MACH principles are useful guidelines for developing composable services, the MACH Alliance is fundamentally a vendor-driven effort with limited interests.

- Many vendors that offer popular API-centric SaaS (such as Elastic Path and Twilio) have not joined the alliance or invested in getting their offerings certified.
- Product teams may be uncertain where and when to apply MACH principles. Not all MACH principles deliver the same level of benefit. API-first is typically very beneficial for all software, while cloud-native SaaS may be unnecessary for internal services. Cloud-native architecture is beneficial if the application requires high scalability and resilience, but the SaaS principle is mainly for vendors or organizations looking to offer software capabilities as public SaaS.

User Recommendations

- Adopt the API-first principle when developing custom software or evaluating solutions.
- Adopt all MACH principles when building externally facing API services. Consider obtaining MACH Alliance certification if the marketing advantages support your business objectives.
- Apply all MACH principles when building internal shared API services to support composable architecture. These services don't need to be deployed in the public cloud to exhibit cloud-native characteristics.
- Evaluate API-centric SaaS and other SaaS-based componentry based on their implementation of MACH principles.
- Consider MACH certification as an optional selection criterion when purchasing digital commerce components, but don't require certification for all API-centric SaaS offerings.
- Explore the MACH Alliance service catalog for MACH-compliant services. Be aware that MACH certification is primarily for marketing purposes by digital commerce vendors.

Sample Vendors

Amplience; BigCommerce; Branding Brand; commercetools; OneStock; Stripe

Gartner Recommended Reading

When Should Applications Use a MACH Architecture Approach?

How to Create Shared API Services to Enable Composability

CSP Open APIs

Analysis By: Amresh Nandan, Ajit Patankar

Benefit Rating: Transformational

Market Penetration: More than 50% of target audience

Maturity: Early mainstream

Definition:

Communications service provider (CSP) open APIs refer to publicly available APIs that allow developers programmatic access to CSP IT and operational technology (OT) applications. 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 network 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, and to foster innovation. 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), CAMARA project 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 compared to other functions, as CSPs have prioritized the use of open APIs in their BSS and OSS application integration.
- There are some key reasons for end-user adoption of open APIs:
  - The desire to achieve greater freedom in sourcing, seamless integration and lower operational costs.
  - The ability to expose certain standard APIs for utilization by enterprises and developers (such as for private mobile network requirements).
  - To drive innovation via developers.
- 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 better management and monetization of the network.
- Some leading CSPs have also started focusing on open APIs for the monetization of data, intelligence and network assets through their service management/digital enablement platforms. We see this focus to intensify in the coming years.
- Some API management solutions, including API gateways, have specific support for CSP APIs such as TM Forum APIs.

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 API 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 API 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.
- Update your procurement process to assess 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 takes time and investments.
- Ensure access to development resources and support the 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-based 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

GraphQL APIs

Analysis By: Andrew Humphreys, Dave Micko

Benefit Rating: Moderate

Market Penetration: 5% to 20% of target audience

Maturity: Adolescent

Definition:

GraphQL APIs provide an abstraction layer to simplify retrieving data from multiple sources. They enable consumers to specify the attributes of data they want to receive. This contrasts with a traditional REST API, where the structure of the data returned is predefined. GraphQL APIs enable efficient, flexible communication between client applications and back-end services, but can create complexity if applied to the wrong use cases.

Why This Is Important

GraphQL APIs gained popularity among software engineers as a standard that allows consumers to use a simple query language for APIs. It enables flexible self-service access to data, particularly for front-end applications. It optimizes data movement between remote clients and back-end services by avoiding overfetching and underfetching issues. GraphQL APIs provide mutation and subscription features that allow API consumers to update data and be notified of server-side state changes.

Business Impact

GraphQL APIs enhance user experience development by enabling access to data based on individual application needs. It is a more efficient use of resources compared to the provider-defined REST API resource model, which has the overhead of individual REST API calls. 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

- GraphQL APIs allow API consumers to define the structure for the responses they require from APIs, enabling greater flexibility than REST APIs. It 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.

- Web developers — particularly React developers — are driving demand for GraphQL APIs for use cases supporting UI and other front-end development.
- GraphQL API use is expanding beyond its original purpose to include back-end, data aggregation and service-to-service use cases.
- 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 require investment in building, managing and operating additional infrastructure beyond typical API management tools. For example, they require a GraphQL server-side runtime for executing queries, implying additional costs, personnel and processes.
- Vendor support for GraphQL APIs, while gradually growing, is still not as widespread as support for REST APIs. Security, governance and performance optimization practices are still maturing.
- Performance is a consideration, as processing a query and efficiently retrieving the specified data 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 that the developer needs to plan for.

User Recommendations

- Consider using GraphQL APIs in your front-end applications to provide flexible access to data, particularly for dashboards and mobile apps. Note they are not well-suited for external consumers due to authorization and rate-limiting difficulties.
- Assess if there is significant demand for GraphQL APIs from your front-end developers, for example, to replace dedicated “experience APIs” or backends for frontends (BFFs), to justify investment.
- Evaluate API management vendor support. GraphQL requires additional infrastructure, such as GraphQL servers and gateways, which is beyond what is typically provided in an API management tool. Determine how GraphQL can fit with your API management and tooling strategy.
- Adopt community-of-practice approaches to mentor developers. 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

Amazon Web Services (AWS); Apollo GraphQL; Escape; Hasura; Hygraph; IBM (StepZen); Inigo Labs; Rapid; Tyk; WSO2

Gartner Recommended Reading

[When to Use GraphQL to Accelerate API Delivery]

[Choosing an API Format: REST Using OAS, GraphQL, gRPC or AsyncAPI]

API Marketplaces in Banking

Analysis By: Don Free, Mark O'Neill

Benefit Rating: Moderate

Market Penetration: 5% to 20% of target audience

Maturity: Adolescent

Definition:

API marketplaces are platforms designed to publish and market APIs. They range from API directories and API portals provided by a single provider to commercial marketplaces. Consumers, mainly developers, use these marketplaces to discover APIs and — in some cases — purchase access to them.

Why This Is Important

API marketplaces enable and empower organizations to share APIs with a community of developers. They help create ecosystems by enabling partners to implement solutions using their APIs. This capability is tied to banks' continued banking as a service (BaaS) and embedded finance revenue objectives. Beyond helping developers discover and share APIs between teams, bank-owned API marketplaces are on the rise to deliver business and corporate banking APIs to clients. As banks embrace advanced business models, such as BaaS, API marketplaces in banking are indispensable.

Business Impact

As API providers, banks can register APIs in marketplaces to increase partnership opportunities with fintechs and other banks. Developers use the banking APIs in API marketplaces, which can increase business impact in the form of improved customer service and increased revenue. Also, API marketplaces are critical enablers of composable business. Ultimately, API marketplace providers may take a share of the revenue for APIs sourced through the marketplace, but this can be considered as a cost of sale for banks.

Drivers

- The number of banking APIs within an organization is climbing, driving the need for developers to find out which APIs and services are available through a marketplace.
- API aggregators also need a source of APIs to fulfill their intermediary roles of speeding and scaling interactions between banks, fintechs and vendors, such as Envestnet Yodlee, MX Technologies, Plaid, Tink and TrueLayer.

- The rise of 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 platform as a service (iPaaS), robotic process automation (RPA) and analytics tooling enables more citizen development, as APIs can be easily sourced from API marketplaces.
- Regulatory influences, such as the U.S. Consumer Financial Protection Bureau (CFPB) Section 1033, are encouraging API adoption within the banking industry.

Obstacles

- **Developer behavior:** Overall, public API marketplaces, which provide a public API directory, had disappointing results because developers are more likely to go to providers directly to sign up for APIs. This has resulted in API marketplaces approaching the Trough of Disillusionment phase. However, internal API marketplaces have had more success, as they enable developers to share APIs across multiple teams.
- **Lack of orchestration:** Disconnected API marketplaces and API developer portals result in slower API adoption, higher support costs and missed growth opportunities.
- **Migration from API developer portals to marketplaces:** API portals provided as part of API management platforms are standard capabilities, but require significant customization. The customization to deliver the look-and-feel, including documentation, getting-started videos and other assets for an API used to describe typical use cases for it, and details on version history and the roadmap as part of an API marketplace.
- **Broader success in BaaS initiatives:** This will remain elusive for midsize and large-tier banks that don't advance API marketplace maturity to enable multipartner, collaborative efforts.

User Recommendations

- Manage senior business stakeholders' expectations by explaining that outcomes from placing APIs in public marketplaces are often disappointing, but that regulatory mandates for open banking and the potential of embedded finance and BaaS will drive adoption.
- Direct your marketplace initiative leader to examine billing terms to understand what goes to the marketplace provider when considering commercial API marketplaces. Since your APIs may be side-by-side with competing APIs, think carefully about differentiation.
- Include technology considerations in the overall business case or commercial model upfront, as well as recommendations for clear governance in onboarding third-party APIs, if your bank plans to use its own API marketplace.
- Use APIs from trusted marketplaces and API service providers, carefully examining usage agreements, licensing and billing terms. In general, ensure that you govern usage of third-party APIs and be clear where data ownership and privacy lie.

- Investigate if subscribing to an API directly from the API provider offers better pricing or usage terms than consuming the API through a marketplace.

Sample Vendors

Crosskey; DigitalAPICraft; Google; Nordea; Pronovix; RapidAPI; Salesforce

Gartner Recommended Reading

Leverage API Portals for Successful API Adoption

To Create a Successful API Ecosystem, Look Before You Leap

Emerging Tech: How to Buy the Right Ecosystem or Marketplace Solution

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

API Threat Protection

Analysis By: Mark O'Neill, Dionisio Zumerle

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. Specialist API threat protection tools, API gateways, and web application and API protection (WAAP) protect APIs by offering visibility, a combination of content inspection of API parameters and payloads, managing traffic, and analyzing traffic for anomaly detection.

Why This Is Important

Organizations use APIs to provide access to data and 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 using techniques such as privilege escalation may result in data breaches, leading to loss of sensitive data as well as reputational harm. A successful DoS attack on an API may result in a lack of availability.

Business Impact

When APIs are typically used for accessing 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

- A persistent stream of publicized API breaches continues to drive awareness of the need for API threat protection.
- Certain industries, including financial services and e-commerce, involve APIs that are high-value by nature, and therefore vulnerable to abuse and fraud.
- APIs are now widely used by generative AI (GenAI) for data access, to provide model-training data, leading to an increased need to protect that access to data.
- Some established vendors have started using machine learning to detect potentially harmful API usage patterns and thus protect APIs from threats. These include WAAP vendors.

Obstacles

- Organizational ownership of API threat protection is a challenge as the security team, under a chief information security officer, typically manages a web application firewall (WAF), while API gateways are typically 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 GenAI, most 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 DoS protection and bot detection. Behind this layer, apply API-specific authentication, authorization and protection mechanisms.
- Assess the API protection provided by your current WAF vendor. 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 Internet Protocol allow/block lists are rarely useful in production environments or at scale.

Sample Vendors

42Crunch; Akamai Technologies; Cequence Security; Cloudflare; F5; FireTail; Imperva; Salt Security; ThreatX; Traceable

Gartner Recommended Reading

Market Guide for API Protection

Research Index: Everything You Should Do to Address API Security

API Security Maturity Model

Innovation Insight for API Protection

API Security Testing

Analysis By: Dale Gardner, Giles Williams

Benefit Rating: High

Market Penetration: More than 50% of target audience

Maturity: Adolescent

Definition:

API security testing is a specialized type of application security testing (AST) that identifies vulnerabilities in APIs. Checks should include traditional application vulnerabilities (such as injection attacks) and API-specific issues (e.g., broken-object-level authorization). Discovery capabilities are sometimes supported to 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. Vulnerability assessment teams may test the security of production APIs; however, APIs pose unique risks.

Business Impact

APIs are a foundational element of many organizations' digital transformation efforts. Hence, securing APIs from attack and misuse is an ongoing 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 can be 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 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 contend with undocumented or shadow APIs, which exist outside normal processes and controls. Such APIs may be 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 — static application security testing (SAST), dynamic application security testing (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. Nor were they aimed at newer types of APIs (such as GraphQL or gRPC). API-specific vulnerabilities and modern API formats prompt security and development teams to implement specialized API security tools focused on testing, discovery of shadow APIs and protection from threats (or a 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, and business logic tests are difficult to automate. Specialized tools or penetration testing may be needed to reliably detect these flaws.

- Testing tools may not support all API protocols. SOAP remains in widespread use, although it is being supplanted by REST APIs. GraphQL-based and gRPC-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, as well as API testing and protection vendors), complicating evaluation and selection efforts.

- Licensing and pricing can be difficult to manage, especially when discovery is lacking.

User Recommendations

- Begin tool evaluation by focusing on the criticality of APIs, their security and business risks and the technical requirements they pose. This will help determine needed functions and the level of testing necessary.

- Assess the suitability of the testing and discovery capabilities provided by tools in your application security portfolio. Many testing tools now include support for APIs, but tests may focus primarily on traditional application vulnerabilities, not those specific to APIs. API security tool capabilities — including discovery, testing and threat protection — often overlap.

- Evaluate dedicated API testing tools 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, because traditional AST is often unable to detect some common API-specific vulnerabilities.

Sample Vendors

42Crunch; Akamai Technologies; Akto; APIsec; Bright Security; Contrast Security; HCLSoftware;; OpenText; PortSwigger; StackHawk

Gartner Recommended Reading

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

How to Deploy and Perform Application Security Testing

Market Guide for Cloud Web Application and API Protection

Magic Quadrant for Application Security Testing

Critical Capabilities for Application Security Testing

Service Mesh

Analysis By: Anne Thomas

Benefit Rating: Low

Market Penetration: 1% to 5% of target audience

Maturity: Adolescent

Definition:

A service mesh is a distributed computing middleware that manages communications between application services — typically within managed container systems. It provides lightweight mediation for service-to-service communications and supports functions such as authentication, authorization, encryption, service discovery, request routing, load balancing, self-healing recovery and service instrumentation.

Why This Is Important

A service mesh is lightweight middleware for managing and monitoring service-to-service (east-west) communications — especially among microservices running in container management systems, such as Kubernetes. It provides visibility into service interactions, enabling proactive operations and faster diagnostics. It automates complex communication concerns, thereby improving developer productivity and ensuring that certain standards and policies are enforced consistently across applications.

Business Impact

Service mesh is one of many middleware technologies that provide software infrastructure for distributed applications. Service meshes are most often used with services deployed in container management systems, such as Kubernetes. This type of middleware, along with other management and security middleware, helps provide a stable environment that supports “Day 2” operations of containerized workloads. However, the technology is complex and often unnecessary for smaller deployments.

Drivers

- **Microservices and containers:** Service mesh adoption is closely aligned with microservices architectures and container management systems like Kubernetes. Service mesh supports useful functionality in ephemeral environments, such as dynamic service discovery and mutual Transport Layer Security (mTLS) between services.
- **Observability:** As microservice deployments scale and grow more complex, DevOps teams need better ways to track operations, anticipate problems and trace errors. Service mesh automatically instruments the services and feeds logs to visualization dashboards.
- **Resilience:** A service mesh implements the various communication stability patterns (including retries, circuit breakers and bulkheads) that enable applications to be more self-healing.
- **Bundled feature:** Many container management systems now include a service mesh, inspiring DevOps teams to use it. The hyperscale cloud vendors provide a service mesh that is also integrated with their other cloud-native services.
- **Federation:** Independent vendors such as Buoyant, Greymatter.io, HashiCorp, Kong and Solo.io provide service meshes that support multiple environments.

Obstacles

- **Not necessary:** Service mesh technology can be useful when deploying microservices in Kubernetes, but it’s never required.
- **Complexity:** It’s complex to use and administer, and there are increasing discussions on why not to use a service mesh in technology discussion groups and social media.
- **Redundant functionality:** Users are confused by the overlap in functionality among service meshes, ingress controllers, API gateways and other API proxies. Management and interoperability among these technologies is still nascent within the vendor community.
- **Overhead:** Service mesh technology consumes resources and typically adds overhead to the interactions it manages. Some vendors now support alternate architectures, such as a shared-agent model to reduce overhead, but this solution reduces the observability benefits.
- **Competition with “free”:** Independent service mesh solutions face challenges from the availability of platform-integrated service meshes from the major cloud and platform providers.

User Recommendations

- Determine whether the value you might get from a service mesh in terms of improved security or observability is worth the increase in complexity and administration of the service mesh. A service mesh becomes more valuable as the number of service-to-service (east-west) interactions increases.
- Favor the service meshes that come integrated with your container management system unless you have a requirement to support a federated model.
- Reduce cross-team friction by assigning service mesh ownership to a cross-functional platform engineering team that solicits input and collaborates with networking, security and development teams.
- Accelerate knowledge transfer and consistent application of security policies by collaborating with infrastructure and operations and security teams that manage existing API gateways and application delivery controllers.

Sample Vendors

Amazon Web Services; Buoyant; Google; HashiCorp; Istio; Kong; Microsoft; Solo.io

API Industry Standards

Analysis By: Andrew Humphreys

Benefit Rating: Moderate

Market Penetration: 1% to 5% 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

API industry standards provide the technical and, in some cases, legal framework for APIs that enable the sharing of data and services in industry verticals such as automotive, insurance, travel or banking. They improve interoperability and make it simpler for third parties to make use of common APIs, opening new business opportunities for API providers. Implementing APIs that adhere to industry standards simplifies adoption, accelerates development and encourages innovation.

Business Impact

API industry standards reduce design and implementation effort by providing templates for implementing APIs commonly required in the industry. When widely adopted, following the

standards for common industry APIs can help open new revenue streams and find new ways to communicate and engage with customers. API industry standards also make it simpler for third parties to use shared data and services across multiple industry members.

Drivers

- Government mandates are driving the use of industry-specific API standards in healthcare (for example, the Office of the National Coordinator for Health Information Technology [ONC] regulations in the U.S.) and banking (for example, the Payment Services Directives 2 and 3 in the EU, and the Open Banking standards in the U.K.).
- API standards in healthcare have opened up access to data and services in a consistent way, enabling developers to deliver quickly new applications that combine data from multiple API providers. For example, the Fast Healthcare Interoperability Resources (FHIR) standard in healthcare simplifies the sharing of a patient's healthcare data among healthcare providers and improves the customer experience.
- API industry 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.
- Other industries, like container shipping with Digital Container Shipping Association (DCSA), travel and insurance are looking to define industry standards for APIs. Defining these standards would replicate the benefits from the improved customer experience, simplified interoperability and increased innovation in banking and healthcare.

Obstacles

- There is a lack of successful initiatives to define API standards in most industries.
- Using industry standards is not always free and might require joining the organization that oversees them, which limits adoption.
- Even in industries where certain standards are mandated, the definitions of the published APIs may differ, presenting an obstacle to interoperability.
- API industry standards are not required in industries with no regulatory requirements or strong interoperability needs. The interoperability benefits require adoption by a critical mass and early adopters may not see benefits initially.
- Vendor support for API industry standards is inadequate, though slowly growing.
- API industry standards are often too generic and complex to tackle the aggregate of all problems they might be applied to. They are not often the best examples of well-designed, consumer-friendly APIs, and may not meet specific needs.

User Recommendations

- Follow API industry standards if you are in banking or healthcare. The well-established standards in these industries will enable you to benefit from the increased interoperability and innovation.
- Use aggregator API services, such as Plaid or TrueLayer in banking, which provide a single API in front of multiple back-end APIs, enabling the adoption of standard APIs without requiring significant rework in back ends.
- 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 in all cases to enable common understanding and interoperability between API providers and consumers. API standards include guidelines for how APIs are described and published.

Gartner Recommended Reading

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

Market Guide for API Portal Platforms

Market Guide for API Gateways

Climbing the Slope

API Testing Services

Analysis By: Jaideep Thyagarajan

Benefit Rating: Moderate

Market Penetration: 20% to 50% of target audience

Maturity: Early mainstream

Definition:

API testing services are a set of outsourced testing activities 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 by the surge in organizationwide API strategies. An API that fails to deliver the expected level of quality, security, reliability and performance can have cascading effects on both the organization producing it and those consuming it. Risks associated with application failure have broader business impacts; thus, 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 team members with a mandate to deliver high-quality, reliable APIs. In recent times, generative AI has played a significant role in API testing by automating the generation of test cases, data and scenarios.
- Organizations increasingly rely on APIs for facilitating end-to-end transactions. Failure in any of the interfaces means a failed transaction, which equates to poor experience and revenue loss.
- 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, preventing issues from becoming larger during GUI testing, thereby protecting 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 in-house testing resources and many times, they may not have the requisite skills.

- 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, design tests that are relevant for APIs, use API testing tools, understand business logic of service and locate real user scenarios.

User Recommendations

- Begin API testing services provider 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

APIContext; Applause; Infosys; Postman; Simplify3X; SmartBear; Stoplight; TestHouse; 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 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. The server 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 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 must add authentication and authorization controls to API process flows and continue to mature and improve these controls.

Business Impact

API access control capabilities are important in several scenarios:

- Developer use cases, offering libraries, out-of-the-box (OOTB) integrations and best-practice examples of API access controls.
- Enabling user-present 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.
- Required for any organization pursuing zero-trust security strategy.

Drivers

- API access control adoption, including OAuth 2.0, has benefited from the popularity of access management (AM) tools, driven by the continued increase in API-delivered versus browser-delivered functionality.
- Organizations' digital transformation efforts and imperative to provide secure API access for partners and customer engagement journeys drive improved API access control strategies.
- The application attack surface has grown substantially with the increased use of exposed APIs.
- There is a lack of advanced IAM capabilities of API gateways (e.g., advanced adaptive access and support for evolving identity standards). This has increased demand for specialized access control for APIs, delivered by AM, externalized authorization management (EAM) and other IAM and API security specialist tools. This demand is also increased by these same requirements for organizations looking to achieve zero-trust strategies at the API layer. Authorization servers also provide strong, agile cryptographic mechanisms for signing tokens.

- Beyond OAuth 2.0 authorization servers, IT leaders need the ability to centralize authentication and authorization services for libraries and OOTB integrations with APIs and API mediators (such as sidecars).
- Demand for developer self-service capabilities is increasing, including support for the device flow, token exchange, introspection and revocation.
- Modern applications and service patterns (e.g., service mesh, REST and GraphQL) 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 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 access control.
- 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 common viewpoint 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.
- Support for appealing cross tooling developer experiences and EAM is still nascent since API mediators don't provide support to integrate with other tools. API access control tooling therefore needs customized integrations for each infrastructure component, making this not in scope for minimum viable products where compliance doesn't require it.

User Recommendations

- Define a comprehensive API access control strategy by broadening tool selection. API access control requires authorization servers, EAMs, secrets managers, API mediators and other in-line proxies. This is delivered by a combination of AM, 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 support a zero-trust approach.

Sample Vendors

Amazon Web Services; Authlete; Curity; IBM; Microsoft; Okta; Ping Identity; SecureAuth (Cloudentity)

Gartner Recommended Reading

Magic Quadrant for Access Management

Critical Capabilities for Access Management

Architect a Modern API Access Control Strategy

Reference Architecture Brief: API Access Control

API Portals

Analysis By: Andrew Humphreys

Benefit Rating: High

Market Penetration: 20% to 50% of target audience

Maturity: Early mainstream

Definition:

An API portal is a platform to publish and share APIs. Consumers, mainly developers, use API portals to discover and learn about APIs and get access to them. API portals can be internal, used for promoting an organization's APIs to their own developers, or external, targeting API consumers outside of the company.

Why This Is Important

Building API-centric solutions requires developers to be able to easily find APIs. API portals are the key tool developers use to discover which APIs an organization offers. Having an API portal is therefore critical to the adoption of an organization's APIs and the success of an API strategy. API portals must offer a quality user experience and make it simple for consumers to find the APIs they need and get access to use them.

Business Impact

API portals increase API visibility and drive their usage. API portals simplify finding APIs and help explain their functionality so developers can decide if they want to use them. Businesses can also use an external portal as a shop window to advertise their APIs and encourage external consumers to use them, enabling monetization. They can also use it as a key tool for establishing a reputation as a provider of good developer tooling and support to API consumers.

Drivers

- As the number of APIs within an organization grows, it drives the need for developers to be able to discover through self-service which APIs and services are available. API portals often also include additional features to simplify onboarding, such as by including key and credential management, generated code samples and sophisticated developer sandbox environments.
- As API use has accelerated and matured, portals have expanded in role to become the central communication point for all involved in API delivery and use.
- For organizations looking to monetize APIs, external portals are the shop window for potential API consumers. They are critical to not just showing what APIs are available but also to helping external business stakeholders understand an API's business value. They can also provide data to help the management of their API usage and costs.

Obstacles

- Poorly implemented portals that do not contain relevant APIs and do not provide a good experience for discovering, understanding and signing up to use them are a barrier to API adoption.
- Internal and external developers have very different expectations, so it is difficult for organizations to address both audiences with a single portal.
- API management platforms generally provide portal capabilities. Gartner clients, however, often report dissatisfaction with their functionality, usability and customization options.
- New open-source platforms, such as Backstage from Spotify, are driving the creation of internal API catalogs as part of internal developer portals, which may be sufficient for your needs as a full portal is probably overkill.
- API portals must evolve to respond to the changing needs of existing and potential API users. They require dedicated time and investment and should not be considered a one-off activity.

User Recommendations

- Maximize the potential of your API portfolio by providing a portal that focuses on your targeted audiences' needs for how the API catalog is presented and how they will discover the portal itself.
- Address different consumers' requirements by implementing separate internal and external API developer portals.
- Select the right technology to implement an API portal by considering all the capabilities needed to support effective and secure API consumer journeys from API discovery to operation of API-consuming applications. Also consider how to integrate with development processes and API gateways and which customization features are required.

- Ensure your portals evolve with changing requirements by managing them as a product, establishing clear portal ownership and roadmap responsibilities.

Sample Vendors

Apiable; APIIDA; APIMatic; apinty; APIwiz; DigitalAPICraft; Moesif; Sensedia; Speakeasy

Gartner Recommended Reading

Leverage API Portals for Successful API Adoption

Innovation Insight for Internal Developer Portals

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 partner API solutions that are 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

- Increased levels of transparency and self-service within API developer portals will drive higher levels of usability promoting mainstream adoption.
- Banks' priorities to minimize complexity, eliminate duplication and reduce total cost of ownership (TCO) require the distributed management of a range of API resources.

- Banks are looking at expanded business models (e.g., banking as a service) and an API developer portal is a foundational technology and a prerequisite for API marketplaces.
- Organizational and process improvements focusing on internal communication and collaboration are driving API developer portals to mainstream adoption.
- Banks are leveraging API developer portal investments by adapting them for use with low-code platforms.

Obstacles

- Low adoption of existing internal developer portals (separate from API portals) can signal decreased trust in the value of API portals.
- Developers are frustrated with complex navigation and various versions of APIs that are common in API portals.
- Inability for banks to get a return on their investment with other roles in the bank.
- 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 providing tools such as API catalogs and analytics to drive efficiency.
- Drive user adoption by publicly recognizing developers and showcasing 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

OpenID Connect

Analysis By: Erik Wahlstrom, Brian Guthrie

Benefit Rating: High

Market Penetration: 20% to 50% of target audience

Maturity: Early mainstream

Definition:

OpenID Connect (OIDC) is an identity federation protocol built on the OAuth 2.0 framework that enables web services to externalize authentication functions. It enables applications (e.g., web-based, mobile and JavaScript) to authenticate human end users, as well as obtain basic profile information over an API.

Why This Is Important

- OIDC lets application owners and developers authenticate humans across websites and applications without having to create, manage and maintain identities.
- With OIDC, you can provide single sign-on (SSO) and reuse enterprise or social accounts to access applications, thereby improving usability, security and privacy.
- OIDC provides crypto-agility, consent management, support for hybrid and multicloud environments, and support for more client types than previously developed federation protocols.

Business Impact

Built on top of the OAuth 2.0 protocol, OIDC offers a flexible, efficient alternative to SAML. Its main benefits are:

- Improving user experience by providing lightweight authentication, authorization and consent management.
- Reducing the data entry burden during user registration with SSO and federation support.
- Providing better support for key discovery and rotation than SAML.
- Efficiently supporting token and API-centric architectures with mobile and single-page applications.

Drivers

- Interest in adopting OIDC continues to grow to replace SAML in terms of preference for new client-facing and enterprise applications. The benefits that OIDC brings to API access controls, privacy regulation, consent management, step-up authentication, compliance and implementation of adaptive access will accelerate its time to plateau.
- Extensions built for OIDC establish a federation of federation services (multilateral federation), which is commonly used in higher education and with select industries such as healthcare, in which a common set of policies is followed.

- OIDC is a way to use a single set of user credentials to access multiple sites, thereby improving usability.
- OIDC has been proven to allow identity interactions to be conducted more seamlessly and with less friction for developers than XML-based standards, such as SAML, or purely proprietary implementations, and with greater security than preceding protocols.
- Finance, government and healthcare institutions can benefit from the increasing work to profile OIDC specifications to support, and be fine-tuned for use by, industry verticals.
- Work is ongoing to extend OIDC to support more use cases. This includes fast trust establishment between OpenID providers and relying parties, support for verifiable credentials, and the establishment of standards to share risk signals.
- OIDC is mature and well-supported. Organizations can find certified solutions through the OpenID Connect Foundation that certifies solutions against server conformance profiles of OIDC.

Obstacles

- The list of SaaS applications supporting OIDC continues to grow; however, it still has smaller market penetration than SAML.
- Developers often underestimate the intricacies of the protocol and build homegrown libraries with “cherry-pick” features from the specification, making implementations unsecure.

User Recommendations

- Give preference to OIDC over SAML. Use OIDC for modern application “greenfield” developments.
  Leverage an access management tool that centralizes adaptive access engines, supports multiple protocols and can translate among protocols, especially between SAML and OIDC, and other proprietary security token formats.
- Use OIDC for human user authentication, not for machines (workloads and devices), which instead should rely on the OAuth 2.0 framework to get tokens.
- Use OIDC instead of proprietary API keys or authentication methods to avoid vendor lock-in and balance security, privacy, usability and scale when building and deploying applications and services.
- Use proven and well-tested, open-source and/or vendor-provided libraries that are up to date and meet the latest security recommendations.
- Don’t misuse the ID token to call APIs. Instead, use the access token and modern authorization frameworks to validate the access token in and behind enterprise API gateways.

Sample Vendors

Authlete; Curity; Gluu; IBM; Microsoft; Okta; Ping Identity; Red Hat; SecureAuth; Thales

Gartner Recommended Reading

Modern Identity: OpenID Connect, OAuth 2.0, JWTs and SCIM 2.0

Magic Quadrant for Access Management

Buyer's Guide for Access Management

IAM Leaders' Guide to Access Management

Reference Architecture Brief: API Access Control

Lightweight API Gateways

Analysis By: Steve Deng

Benefit Rating: Low

Market Penetration: 20% to 50% of target audience

Maturity: Early mainstream

Definition:

A lightweight API gateway, sometimes called a microgateway, is designed to be distributed and deployed close to individual services. Unlike traditional enterprise API gateways, which are often centrally managed as shared middleware and managed by a central team, lightweight gateways are deployed, configured and managed on a per-application basis, often by development teams.

Why This Is Important

Modern distributed architecture patterns rely on APIs for service-to-service communication. Traditional enterprise gateways have significant management overhead and are best deployed as centrally managed middleware. Lightweight gateways provide essential functionality with minimal overhead.

Business Impact

Lightweight API gateways meet today's enterprise demands for last-mile API security, intelligent traffic management and distributed observability. They allow developers to incorporate API gateway capabilities as part of their application delivery. Teams utilizing lightweight API gateways can deploy new APIs faster compared to those dependent on centrally managed enterprise gateways. Lightweight gateways also support zero-trust approaches by providing access control at the application level.

Drivers

- Diverse API use cases, from traditional API-based integration and multiexperience architecture to emerging generative-AI-driven applications, drive the demands for a new generation of scalable, developer-oriented and highly configurable API gateways.

- Organizations want to control internal API traffic within their trusted network for security and compliance reasons.
- Modern container platforms drive the need for lightweight API gateways to manage ingress, load balancing and service mesh capabilities, especially in a Kubernetes environment.
- Developers building and consuming APIs want to incorporate API management features as part of their applications.
- Centralized API management acts as a bottleneck to the delivery of APIs. For organizations that can't tolerate delays, lightweight API gateways offer a better alternative than no gateway at all.

Obstacles

- The distributed deployment model of lightweight gateways demands more operational maturity and responsibility from the individual product team to manage their own API gateway instances.
- Organizations must transition from a centralized API governance model to a decentralized one with shared responsibility between a centralized team, such as an API platform team and development/product teams.
- Organizations must mitigate the risks of API gateway sprawl brought about by the "bring your own gateway" trend by providing prevalidated lightweight API gateway usage patterns to development teams.
- Consistency of policies for security, access control and governance is harder to achieve with lightweight gateways. Multiple teams must make the same change in their individual lightweight gateway instances.
- Toolings to manage federated API gateways across hybrid and multicloud environments remain limited and immature to match the cohesiveness and consistency offered by single-vendor solutions.

User Recommendations

- Deploy a combination of enterprise API gateways, lightweight API gateways, ingress gateways and service mesh, where appropriate, to meet your organization's need for north-south and east-west communications for APIs and services.
- Choose lightweight API gateways to provide last-mile security, traffic management, fine-grained policy enforcement and observability, particularly for internal-facing APIs in private clouds or on-premises data centers.
- Customize your own lightweight API gateways to govern internal APIs or domain-specific APIs for better isolation, fine-grained security control and DevOps automation.
- Favor lightweight API gateways that support containerization, declarative configuration, CI/CD integration and separation of concern for platform engineering, API administrator and developer

roles.

Sample Vendors

Apache Software Foundation; Envoy Gateway; F5; Kong; Microsoft; MuleSoft; Solo.io; Spring Cloud Gateway; Zuplo

Gartner Recommended Reading

Decision Point for Mediating API and Microservices Communication

How to Evaluate API Management Solutions

Market Guide for API Gateways

Market Guide for Service Mesh

Reference Architecture Brief: API Management

Backend for Frontend

Analysis By: Anne Thomas

Benefit Rating: Moderate

Market Penetration: 20% to 50% of target audience

Maturity: Early mainstream

Definition:

The backend for frontend (BFF) pattern prescribes creating a dedicated back-end API for each front-end experience. Each BFF provides an optimized API that supports the specific needs of its associated front-end experience, and it encapsulates any orchestration or navigation of back-end services required to support the front-end's workflow. It may also provide app-specific data or logic.

Why This Is Important

The BFF pattern is a popular pattern for enabling multiexperience, which provides optimized experiences for different personas interfacing through different types of devices and modalities. The BFF pattern is often combined with the micro front-end (MFE) pattern to more easily enable reuse and increase team autonomy. These patterns provide more flexibility, higher optimization and a better developer experience than using a single API to support multiple experiences.

Business Impact

The BFF pattern gives front-end teams control over their APIs and reduces risks associated with shared APIs, such as dependency delays and unintended side effects from back-end changes. It simplifies front-end development and enables faster iterations, thereby accelerating delivery of new features. It can improve app performance by optimizing network traffic between front ends and back ends, and it can improve user satisfaction by simplifying the application's workflow.

Drivers

- **Multiexperience:** Product teams frequently need to implement multiple front-end experiences for an application to support different types of devices and modalities, as well as different user personas. Each user experience often requires unique functionality, which places a burden on the back-end services to support multiple data models and workflows. The BFF pattern creates a dedicated API for each front end.
- **Simplicity:** Product teams often implement back-end functionality as a suite of services, which places a burden on front-end developers to implement complex workflow and navigation logic to invoke the appropriate services in the correct sequence to accomplish a task. The BFF pattern shifts the complex logic and workflow to a BFF service, thereby simplifying the front end.
- **Performance:** Product teams may have requirements to reduce network traffic for specific front ends, such as smartphones or watches. The BFF pattern enables product teams to design optimized APIs that can reduce network traffic and latency and improve overall app performance. Also, by transferring the orchestration logic and workflow to a BFF service, the front end can accomplish its work with fewer API calls.
- **Team autonomy:** Front-end components are dependent on back-end components, and when those components are developed by different product teams, diverging team schedules can cause frustrating delays. When using the BFF pattern, front-end product teams typically are also responsible for building their BFF APIs and associated services, which alleviates many, but not all, scheduling challenges.

Obstacles

Every design pattern involves trade-offs that discourage developers from adopting it. The BFF trade-offs include:

- **API proliferation:** Building a separate API for every front end will result in a huge number of APIs and services, each of which must be secured, tracked and managed.
- **Duplication:** BFF services for a particular application often contain similar and redundant logic and back-end service aggregation functionality.
- **Overloading:** In an effort to avoid duplication, product teams may try to pack more functionality than appropriate into a BFF service, or they may shift some of the functionality to the front end.

User Recommendations

- Build separate BFF APIs to support front ends that have significantly different requirements, although also consider alternative patterns and technologies, such as GraphQL, which can efficiently support multiple front ends using a single back-end service.
- Combine the BFF pattern with the MFE pattern for complex front-end apps to enable more agility via independent deployments of new functionality, although beware of the increase in complexity resulting from these patterns.

- Avoid creating a separate BFF service for every front end to mitigate API proliferation. When two or more BFF APIs expose essentially the same functionality, those front ends should be able to share a single API.

Gartner Recommended Reading

How to Make the Right Technology and Architecture Choices for Front-End Development

Adopt a Mesh App and Service Architecture to Power Your Digital Business

When to Use GraphQL to Accelerate API Delivery

Entering the Plateau

API Marketplaces

Analysis By: Andrew Humphreys

Benefit Rating: Moderate

Market Penetration: 20% to 50% of target audience

Maturity: Adolescent

Definition:

An API marketplace is a place to discover and share APIs. Marketplaces differ from company-specific API portals in that they provide APIs from multiple sources so developers can access a wide range of APIs offered by different providers and, in some cases, may purchase access to them.

Why This Is Important

API marketplaces enable organizations to publicize their APIs alongside APIs from multiple other providers. This allows developers to explore APIs from multiple sources in one place, simplifying discovery and registration of APIs. Many API marketplaces also offer monetization options, enabling API providers to generate revenue by charging developers for API usage.

Business Impact

API marketplaces can increase developer visibility and drive API usage, and, by extension, increase business impact. API consumers can use marketplaces to simplify discovering 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 monetization.

Drivers

- Developers looking to build applications using external APIs to provide access to data and key functionality require the ability to easily find and use them. Marketplaces provide a single place where developers can discover APIs they may want to use from multiple providers.

- Organizations that make their APIs available externally should advertise what APIs they have and describe their functionality. If they are looking to monetize by charging to use the APIs, they also need billing and monetization capabilities. An API marketplace can simplify this by providing billing and monetization features through the marketplace so the organization does not have to implement them.

- API providers looking to grow the use of their APIs externally need access to the ecosystem of external developers who are potential consumers of their APIs. Marketplaces that offer APIs targeting similar developer requirements, industries and capabilities are common places for developers to search for APIs and to provide access to that ecosystem.

- API providers with their own portal can make it a marketplace, allowing other providers to add their APIs. This can increase usage as developers may find more choice, and can provide a revenue stream as the other providers can be charged to add their API and to have people subscribe to it.

Obstacles

- API portals specific to a single provider have got greater traction and are a more common way for businesses to expose their API catalog to external developers than using an API marketplace.

- Public API marketplaces that provide a public directory of APIs from multiple providers have had disappointing results. API vendors find it challenging to stand out in API marketplaces that are saturated with poor-quality APIs, making it difficult for providers to get their API discovered.

- API marketplaces often do not give developers the data they need to make decisions about their API, making it difficult for an API developer to understand how their API is being used and so improve it.

- API marketplaces typically charge providers a fee for each API call made to an API that is signed up to be used from the marketplace. This can significantly reduce the revenue the API provider receives.

User Recommendations

API providers should:

- Create an internal API portal, focused on the needs of software engineers to share APIs across the organization.

- Determine the usage of APIs among their target audience before sharing their APIs outside the organization and examine billing terms to understand the cost and benefit of using commercial API marketplaces.

API consumers should:

- 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, usage terms and support than consuming via an API marketplace.

Sample Vendors

Achieve Internet; Bump.sh; Postman; Pronovix; ReadMe; SmartBear; Spotify

Gartner Recommended Reading

Innovation Insight for Internal Developer Portals

Reference Model for API Management Solutions

Appendixes

See the previous Hype Cycle: Hype Cycle for APIs, 2023

Hype Cycle Phases, Benefit Ratings and Maturity Levels

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.

Source: Gartner (July 2024)

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 2024)

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 2024)

Evidence

[1] **2024 Gartner API Strategy Survey.** This survey was conducted online from 27 February through 8 March 2024 to understand the API strategy of organizations through API usage, API styles and AI APIs. In total, 89 IT leaders who were members of Gartner's Research Circle, a Gartner-managed panel, participated. The respondents were screened based on their knowledge about the use and priorities of APIs in their organizations. They were primarily from North America (n = 43), EMEA (n = 33), Asia/Pacific (n = 10) and Latin America (n = 3). Disclaimer: The 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.

[2] **Gartner Software Engineering Survey for 2024.** This survey was conducted to identify the most important roles and skills for software engineering leaders and the change in their demand and importance since last year, understand how talent is sourced generally and for acquiring necessary artificial intelligence (AI)/machine learning (ML) skills, and what tools are seen to increase developer productivity and the metrics used to measure them. It also examined how software engineering leaders anticipate change in their operating budgets and the cost management steps taken. It further aimed to identify the quality and testing techniques and programming languages that software engineering leaders currently use and/or plan to use; their frequency of usage of UX design, user research and AI in generating components of user experience; and its impact on user satisfaction, accessibility and usability. It also intended to understand the software engineering leaders' responsibilities they find most difficult, the career paths available for senior-level individual contributors and how they are set up, how organizations attract and retain top performers in those career paths, and what management training is offered to staff. The survey was conducted online from October through December 2023 among 300 respondents from the U.S. (n = 241) and U.K. (n = 59). Qualifying organizations operated in multiple industries (excluding the IT software industry and education sector) and reported enterprisewide revenue for fiscal year 2022 of at least $250 million or equivalent, with 63% 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 study do not represent global findings or the market as a whole but reflect sentiment of the respondents and companies surveyed.