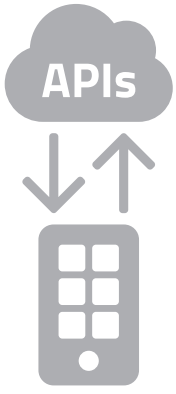
API Management and CIAM – The New Dream Team for Digital Business Models?

Digital space offers ideal conditions for transforming classic business models, but also for developing completely new approaches. The prerequisite for this is reliable, secure, and convenient identity management. From the user's perspective, the services offered are even more important. The combination of CIAM and API management promises to enable users to combine them according to their own needs, to create their own digital ecosystem, and to switch seamlessly between different apps, services, and providers with one identity. In this white paper, we explain how this offers enormous advantages and potential to providers, as well.

As early as 2006, British entrepreneur Clive Humby formulated the slogan "Data is the new oil". Even before the market launch of the first iPhone, the claims of the digital industry were thus redefined. Since then, the volume of data has grown exponentially: in ever-expanding online commerce, in the ever broader use of ever more diverse social media platforms and, last but not least, in the B2B sector, where the availability and analysis of corporate data play a decisive role in economic success. Nevertheless, the question remains: If data is the new oil, where is the filling station? Where does the raw material get turned into a suitably processed, consumable resource? The answer is simple: at the interfaces between users and service providers, at the APIs.



Today, most apps no longer run only on the local device, but are tightly coupled with APIs that are available in the cloud.

A Quick Look Back

From the perspective of computer science, APIs are not a new idea. Ultimately, it is about distributed systems. Approaches already existed in the 1990s with CORBA, but this was technically very complex. In the 2000s, the focus shifted more to service-oriented architectures and their implementation, using Java Enterprise Edition, SOAP, or DCOM. In general, these API approaches still involved server-side business processes. The first web APIs were developed as part of AJAX architectures in the context of dynamic websites and then gained massively in importance with mobile apps towards the end of the decade. The change from desktop systems to mobile computing was accompanied by a rapid development of web ecosystems. Today, most apps no longer run only on the local device, but are tightly coupled with APIs that are available in the cloud. In fact, many apps generate their benefits by combining services from different providers under one interface, thus creating added value: for instance, when a tourism app combines hotel booking with map services and other services. Depending on the

Procedures were needed to manage the APIs, regulate access, and bring order to the whole thing. This is exactly what API management does.

form of travel, customers then receive, for example, an individual route or suitable car rental offers.

From Order to Ecosystem

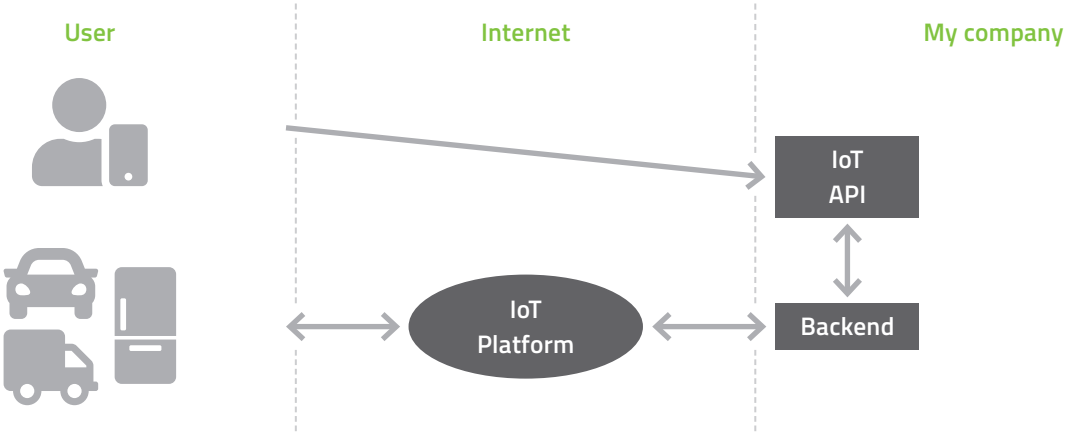
The number of APIs exploded in the course of this enormously rapid spread. Both users and developers were overwhelmed. Procedures were needed to manage the APIs, regulate access, and bring order to the whole thing. This is exactly what API management does. It clearly defines to whom which data is made available, and how. Whether the request comes from the user or from another API is irrelevant. On the contrary, this is the reason why digital services are no longer just side-by-side applications, but, by providing APIs, have the potential to become a real digital ecosystem.

One vivid example is "If This Then That" (IFTTT). This service combines events with actions according to a simple if-then logic. This way, it connects solutions from providers without them knowing about each other. For example, a digital door lock can provide locking or unlocking as an event via an API. The alarm system from another manufacturer may perform certain actions (arming, disarming), also via an API. Using two simple rules, IFTTT can now integrate these two devices at a higher level: If the door lock is locked, activate the alarm system; if it is unlocked, deactivate it. There is no business relationship between the supplier of the door lock and the alarm system – they probably don't even know that the other exists.

Another example of how different services can be combined to create something entirely new are digital assistants such as Alexa and Siri. They enable playback of the current Top Ten via subscribed streaming services, to voice-controlled lighting and air conditioning. With the help of such services, users can create a digital environment entirely according to their own needs. Providers, on the other hand, benefit from significantly stronger customer loyalty. While it is possible to replace one solitary app with another without much effort, the effort increases with increasing networking.

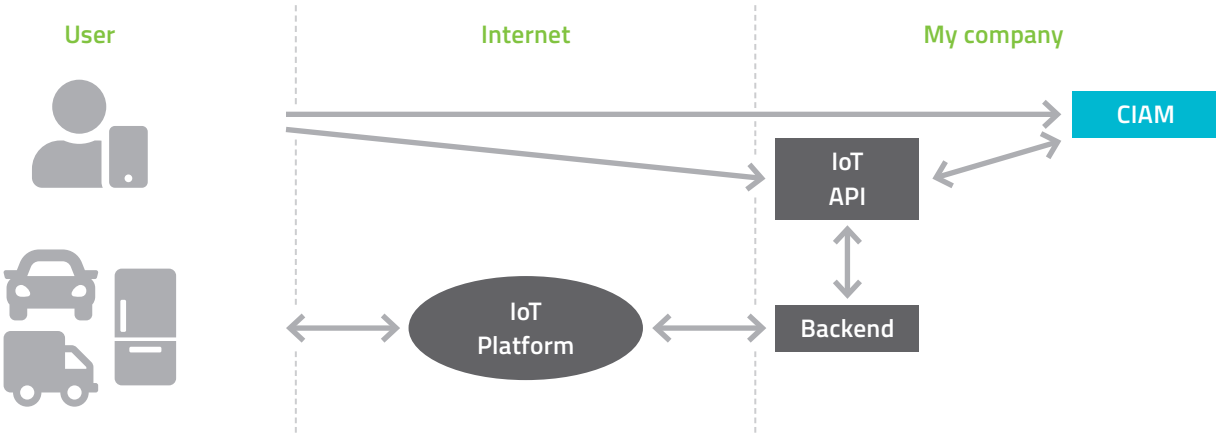
CIAM Comes into Play

Many APIs also work well without identity verification. For example, when requesting public information such as opening hours, gas stations displayed according to price, bus and train delay reports, personal registration is usually not required. All that is needed is the provision of an API that communicates with the IoT devices via a backend. The user's identity does not yet play a role at this first evolutionary stage of the digital ecosystem. However, this approach quickly reaches its limits. For example, if several people use a car, the desire to create their own profiles will quickly arise, even when querying public information. Not for reasons of data security, but for ease of use.



Simple API scenario without CIAM

The situation is different if additional services are used for a fee. For example, if the washing machine is supposed to reorder detergent on its own. Or in the case of shared–use appliances. For example, it makes perfect sense not to share the stored social media profiles or news preferences when sharing a car.

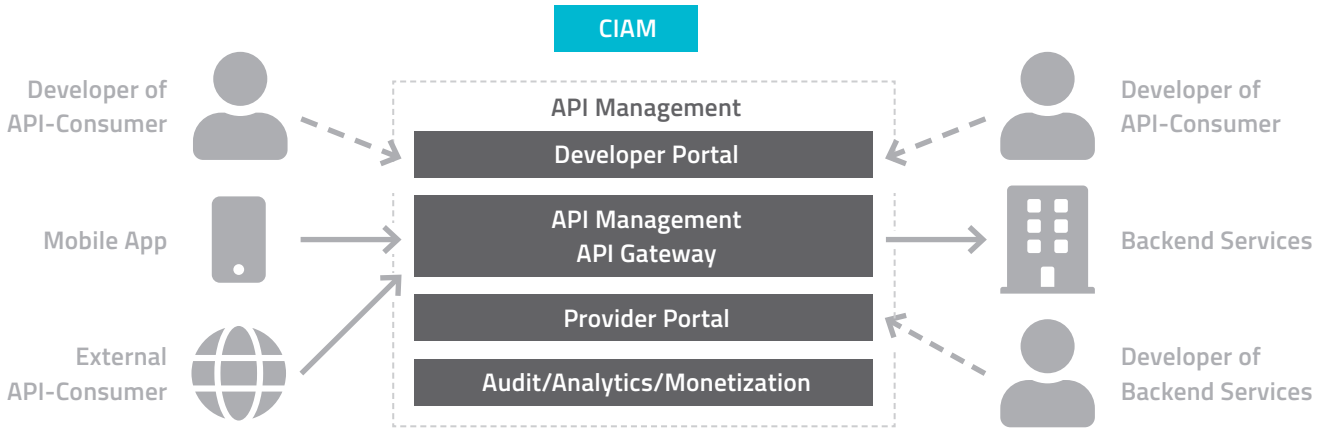


Even if several users with individual profiles jointly operate IoT devices, identity management is necessary.

The scenarios discussed so far can also be implemented without API management. However, if additional services with their own APIs come into play, things quickly become confusing. With each additional API, the effort for administration, configuration, and integration increases exponentially. Not to mention the associated security issues. Things can become overwhelming. API management restores the overview and greatly simplifies building a digital ecosystem for developers. Here, developers receive all the information they need to call the API. This is not only about documenting the access methods, but also about applying for, activating, and providing passwords for access to the APIs. Of course, this is not about giving all users access to all APIs. The necessary consent of the end user is obtained and managed by the CIAM.

A Look Behind the Scenes – Structure and Components of an API Management Setup

API Management – Components and Roles



To understand the function of an API management setup, it helps to take a closer look at the components involved.

First of all, there is the API gateway. It makes the provider's backend services available to apps and other API consumers in a secure manner.

The provider portal is particularly important for the provider. It gives the developers of backend services the functions needed to provide their services. Here, the backend services are connected to the API and, if required, usage costs are determined. In addition, the extent of API usage by developers can be tracked here and insight gained into any revenues that may be generated from this.

For users of the APIs, primarily app developers, the developer portal is the central point of contact. Here, they can see which APIs are available and how they are called. In addition, app developers can register their apps for access to the API. Of course, this raises the question of access rights to the developer portal. Experience shows that most developer portals start as an in-house solution. Simply out of the desire to make already developed tools available to their own employees and to avoid multiple developments of the same functionality. Relatively quickly, it then becomes apparent that partner companies and suppliers also benefit from the developer portal. In fact, it is recommended to open it to independent third parties. More on this below.

But isn't it risky to grant access to company functions to completely unknown people? Every API exposed to the Internet increases the attack surface – but this is also the case without API management and provision for third parties. Apps can be decompiled or eavesdropped on to determine how an API works. Calling an API cannot be reliably prevented with reasonable effort. Instead of investing unnecessary energy in usage restrictions, it is usually much more efficient to define rules for usage and to provide

documentation. In addition, when providing an API – whether public or only for a specific app – one thing must be clear: All security controls are done in the backend. No app is ever considered trustworthy.

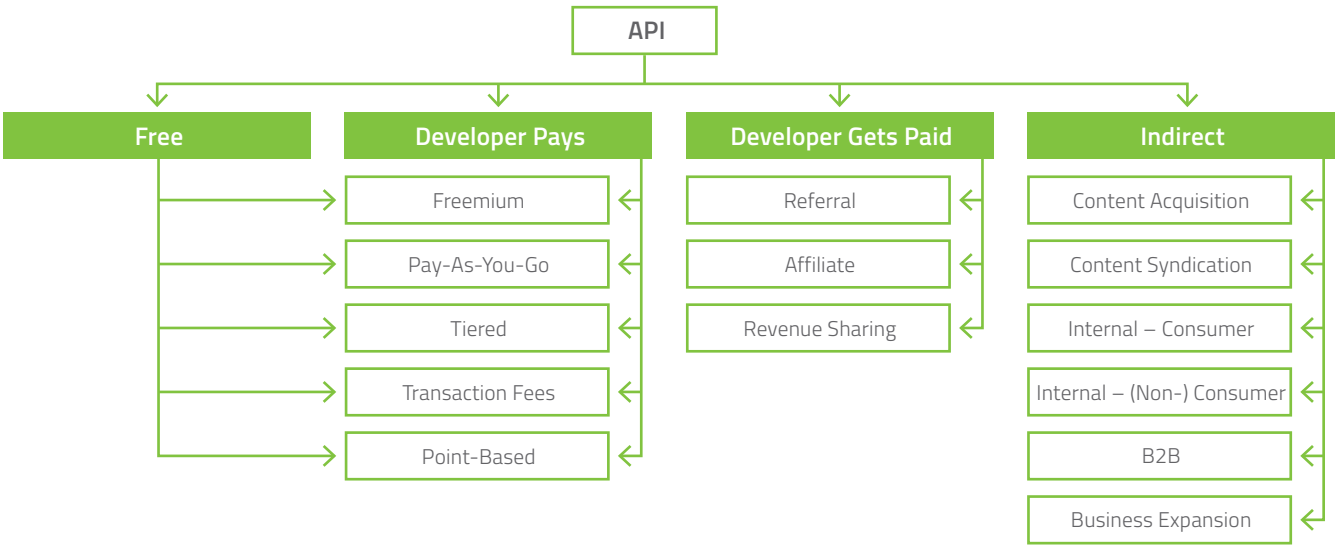
If you like, you could also say: Instead of raising high walls to make the view over them all the more interesting, it would make more sense to set up a route plan and, if necessary, charge admission fees for direct viewing.

Even if monetization is not yet a major issue at present, it is important to consider when exploring API management. Because APIs are an ideal tool for making a company's digital assets usable and establishing new business models. Before doing so, however, three fundamental questions should be clarified:

- What benefits does the API offer potential customers?
- Who are the potential customers (partners, companies, independent developers)?
- How can they get the most out of the API itself?

Especially the last point is crucial for choosing the appropriate monetization model: free, developer pays, developer is paid, or indirect models.

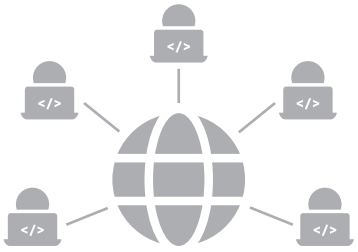
API Monetization Models and Variations



Depending on the digital asset, there are various marketing possibilities. Freemium models, i.e. the free provision of basic functions, allow customers to convince themselves of the benefits of the application. Pay-as-you-go is always suitable if the API is used irregularly and only when needed. Tiered models are similar to volume tariffs, where the customer orders a certain number of API calls per month, for example. If the quota is exceeded, access can be blocked, the price increased, or an additional package booked. The point-based approach works similarly. The customer buys a certain number of points, which is a kind of currency for API calls. In this way, different

operations can be flexibly priced. For example, a GET operation could cost one point, whereas a POST operation could cost two points. Similar flexibility can be applied to other monetization options, where developers are remunerated through affiliate programs, or different cost centers within a company pay for usage.

No matter which option is ultimately chosen: The model must fit the product. It should be transparent and easy to understand. And it must be supported by appropriate monetization functions in API management. This includes the flexible definition of rules for permitted transactions. Monitoring functions monitor the individual actions and make sure that no limits are exceeded. And, last but not least, it is important to provide customers with the best possible support. For example, via the developer portal, which provides them with the tools and information they need.



Those who give these developers access to their own services multiply their chances.

On to New Target Groups – Opening the APIs to Independent Developers

In connection with the Developer Portal, we already pointed out that third parties may also have access to it. In fact, this is a strategic advantage of API management. To do this, you only have to consider how many developers there are worldwide. Those who give these developers access to their own services multiply their chances of becoming part of the digital ecosystem of users and, on top of that, of reaching new target groups. Without development costs, solutions can be created that have not even been thought of in your own company. Even international roll-outs can often be carried out more efficiently with flexible, local development teams than with a large central office that is unfamiliar with local conditions. API management ensures that the intended rules are adhered to.

CIAM and API Management – The New Digital Dream Team

CIAM solutions and API management are not necessarily dependent on each other. They can also be operated separately from each other. However, as more and more use cases involve digital identities in one form or another, both approaches complement each other in a fantastic way.

API management does not have to take care of identity clarification. Since APIs are executed in the context of the customer, this is not even necessary. An API for querying stock prices is not critical in itself. It only becomes security-relevant when, for example, the value of a personal portfolio is to be determined. However, the identity and access management required for this does not have to be provided by the API, but can be carried out independently of it. Moreover, APIs are usually not offered on their own, but run parallel to web applications, e-commerce systems, etc., which require access management anyway. In addition, most systems today are distributed – both on-premises and at various cloud providers. With a central CIAM and uniform token allocation, the APIs can be used conveniently in different worlds. Otherwise, the app developer may have to register multiple times in different API management systems with different rules. In the worst case, this can lead to the end user having to log in multiple times to operate an app. With the support of social logins and independent identity providers, this is made much easier and more user-friendly with your own CIAM. The same applies to the central solution for all aspects of user consent.

The provided functionality can be invoiced in a transparent way.

Conversely, API management can also support the CIAM very effectively. This begins with the integration of applications. Sometimes several dozen, sometimes more than 100 applications run on a CIAM. While CIAM systems are not always good at integrating such applications efficiently, API management was created precisely for this task. Then there is application management, which provides a compact overview of the existing applications and their use.

The monetization functions of API management are another very interesting aspect. They offer the possibility of passing the costs of CIAM on to the applications. Especially in companies needing to implement the arm's length principle, the provided functionality can be invoiced in a transparent way.

Stronger Together

In this brief review, we have seen that CIAM and API management complement each other perfectly on many levels. Both systems are capable of sustainably increasing the efficiency of the other, simplifying processes and, what's more, developing new business models and reaching new target groups. It will certainly stay exciting to follow these developments further.

3 Points You Should Consider When Implementing Your API Management

Developer Portals

Design your developer portal as intuitive as possible. Ensure comprehensive, understandable documentation. Make it as easy as possible for third parties to use. The more convenient access to the API is, the more can be achieved by self-service, the more developers will use it.

Beyond the EU-GDPR

Offer your customers more than statutory compliance. Everyone offers the required minimum. Additional transparency and privacy, on the other hand, is perceived by customers as a unique selling proposition.

Failsafe to Use

The easier and safer an API is to use ("failsafe to use"), the faster and more deeply it will become integrated in digital ecosystems, thus opening up new target groups and business opportunities.