www.pseconsulting.com
+44 (0)20 3771 8520

Blog

By Chris Jones

19th December 2024

Is it time to retire the Payments API?

Billions have been made by building, deploying and supporting payments API infrastructure. However, has the time come to retire customer facing APIs and move onto something better?

The near ubiquitous presence of APIs in payments has been transformative. It has enabled significant more flexibility in the way payment users interact with the payments ecosystem. APIs first emerged to manage authorisation messages for cards, but they can now be found across the lifecycle of payment products from merchant onboarding to reporting as well as a whole ecosystem of adjacent products from lending to issuing and fraud. APIs infrastructure has underpinned the emergence of the current breed of intermediary platforms for embedded finance from PayFacs to SaaS/ISVs because of their security, flexibility and richness.

However, APIs have a number of significant drawbacks for both merchants and platforms (SaaS, marketplaces, etc.) users:

BUILD

- They create a significant **build requirement** for users to create the required end-user UX.
- We hear our SaaS clients talk of $1-3m spend to integrate with onboarding, reporting and support APIs which is sufficient to put off even larger platforms.

PROTECT

- They generate a substantial **data protection burden**for users because information needs to be collected before being passed on.
- This creates issues not only with PCI DSS but also sensitive personal data (like passport details) collected during onboarding

COMPLY

- They place a substantial **compliance burden** on users to create the customer messaging to meet the needs of local/regional regulators (such as Treating Customer Fairly, Customer Duty, PSR price comparison boxes in the UK).
- This places a commensurate burden on regulated entities (PayFacs, acquirers, banks etc.) to audit platforms to ensure they are compliant.

It's particularly the last issue that the payments industry will need to watch closely (the other two issues have been around for a while). Regulators have already expressed their displeasure with the BaaS world in both the US and Europe and its reliance on third party compliance programmes. As embedded finance gets more popular, this issue of layering is likely to grow in importance. APIs are not really fit for purpose to support the growth of the embedded finance model, particularly in regions with proactive regulators with strong compliance mandates.

Given the limitations outlined above, is there a better alternative? History provides a strong clue by looking back at what happened to the original payment API, the authorisation request. Initially merchants had a choice between integrating into acquirer's APIs or using (rather clunky) hosted

payment pages. More recently acquirers began offering hosted payment fields (AKA drop-in UI, components, elements, etc.). These break out core elements of the payments page and allow merchants to customise the location and look and feel (font, colour, etc.) of fields while the data within them is still controlled by the acquirer. This integration approach is now the dominant model for merchants who want to control their check-out experience.

So is this a model for other payment APIs, particularly in the increasingly popular embedded finance world? First, let's look at the benefits to merchant/platform users…

Faster time to market

- Pre-built components allow users to deploy using parameters rather than code
- New products and services from the PSP can be rapidly released using pre-tested UX

Lower integration cost

- The number of engineers/teams required and the level of technical expertise is much lower
- Again, this is true for both the set-up of the service, and for ongoing enhancements

Lower ongoing compliance effort

- There is no need for the PSP to audit the merchant/platform as they are in control of the UX. These lower costs can be passed onto users
- If the platform is regulated themselves (such as a PayFac in Europe) they also know the services should meet

…and there are also several benefits to PSPs themselves:

Faster time to revenue

- Lower upfront costs make the business case to implement easier. This is particularly true in embedded finance where the service is optional, and upfront costs are a major issue
- Faster set-up times reduce the time between contracting and first transaction and therefore first revenue

Better PSP value add

- PSPs are clearing providing a more complete technical solution which can be demoed in sales pitches
- PSPs can use their large data sets, and machine learning models, to optimise the merchant/buyer experience – as they already do in the consumer check-out world.
- There is also reduced risk of intermediation by third parties, such as orchestration platforms, as it is harder to add value when abstracting these more complex services

Full PSP control

- As the PSP creates and manages the UX, there is a much reduced need to carry out audits, keeping the regulators at bay.
- New regulatory requirements can be implemented once, and roll-out at scale, again with minimal audit needs

We already see innovative players offering this service in embedded finance for a wider range of use cases beyond authorisation, e.g. onboarding, reporting, support. This includes early offerings from Stripe and a number of specialist players such as UniPaaS and Rainforest.

The value of drop-in elements is particularly clear in the more complex use cases which PSPs are starting to support, such as where marketplaces or platforms are onboarding merchants across multiple countries as documents, languages and types of business all differ by market. Building a dynamic UI to capture and pass the correct data elements to onboarding APIs, and deal with the many possible exception conditions, is a lot of work for these marketplaces/platforms. It's far better to let the PSP develop the UI once and simply embed it. As well as the set-up benefits, a centrally managed service means changes to compliance requirements can be implemented once by the PSP and automatically appear in all users' UX.

So does this mean the payment API will disappear completely? I would argue that for all but the very largest players, the drop-in UI model will gradually displace the API for many customer–facing scenarios, as it has for the authorisation request. However, there are still likely to be back–office processes where the raw API (inc. webhooks) still has value, e.g. transaction lifecycle status, or integrating reporting data into a data lake. Equally, fully hosted pages are still going to have relevance for smaller users who prefer a plug–and–play service. It's therefore likely that payment providers will need to keep APIs alongside drop-in components, and fully hosted pages, for some time. The API is not quite ready for retirement, but it is going to have to share the stage with a more attractive, flexible co-star...