So, You Want to Build the Next Amazing Marketplace?

Top five Payment API must haves for Payment Facilitators

LEARN MORE

If you're familiar with Payment Facilitation chances are that you're already pretty savvy about payments. Payment Facilitation can be critical in unlocking new business models and enabling growth, but for software developers there are many issues that complicate application design. Developers need to think about not only processing payments, but deal with a range of complex issues associated with managing a community of merchants.

As reminder, a Payment Facilitator (or PayFac®) is an entity that acts as an aggregator, providing payment related services and facilitating transactions for a number of sub-merchants. PayFacs basically sponsor merchants, enabling these sub-merchants to be boarded with the support of a payment processor and underwriting bank. This arrangement provides several efficiencies. It simplifies the process of engaging and signing sub-merchant partners, and puts the PayFac more firmly in control of their business. Instead of your merchants signing up with a third party payment gateway (eroding your margins and control) they in effect sign up with you, and you facilitate payments for your own community of sub-merchants. In essence, you become a mini acquirer.

Operating a PayFac at scale depends on efficient management of functions beyond simple processing like boarding, management, funding, and handling chargebacks. As with most things in software, doing these well demands not only the right infrastructure, but modern, flexible programming interfaces.

Boarding & Sub-merchant Vetting

While not as onerous as establishing a merchant account, whenever you sign a sub-merchant, you and your sponsor bank of are taking on some level of risk. Few PayFacs can afford the administrative overhead of processing new sub-merchant requests manually, so having automation behind merchant setup & vetting is a first essential requirement. PayFac management platforms should offer a boarding API to support streamlined processing and fast approval of submerchants for underwriting. The boarding API needs to differentiate between legal entities and individual sub-merchants attached to those entities (because your sub-merchant may operate multiple stores for example).

Ideally, the API should allow you to pass key information about a legal entity (type of business, tax identification numbers, details on principals, years in business etc.) and receive information back. Is the business legit? Is the address valid? Are outstanding liens? Are there criminal records or a history of bankruptcies? Depending on your business, you'd like to have the option of performing background checks with different levels or thoroughness. In cases where sufficient information cannot be verified electronically, you'd like your payment processor to undertake a manual review on your behalf and notify you electronically of the results so that you receive either an approval or a clear explanation of why a particular application cannot be approved.

vantiv®
smarter/faster/easier/payments™

Once legal entities are approved, you should be able to easily manage and maintain details about individual sub-merchants including banking information, approved transaction limits and the like. To operate as a PayFac you will have registered with the card brands in their respective programs, so the API for boarding sub-merchants should automatically provide notifications as you add or retire sub-merchants.

Transaction Tagging

This will be obvious to most developers, but once you board a merchant, you'll need to be able to tag transactions like Authorizations, Captures and Reversals to the appropriate merchant in your code. This requires that your transaction-oriented APIs be PayFac aware as well. Your business might involve a partner branded web presence or a mobile app that end customers download from Apple's AppStore or Google Play. You're clearly not going to build separate apps or infrastructure for each sub-merchant, rather you'll parameterize your code to deal with multiple sub-merchants. Ideally, this transaction-level tagging should be flexible enough to allow for additional metadata like campaign identifiers, affiliates, or other information useful for downstream reporting. As your business grows, you might provide financial incentives to your merchants by paying them differentially based on their performance around specific marketing campaigns or programs. It's essential that your transaction level APIs and reporting systems support this kind of flexibility.

Chargeback Management

Chargebacks are a headache for any merchant, so imagine the challenge when handling hundreds or perhaps even thousands of sub-merchants. This is another case where the scale of the challenge simply demands a comprehensive API. While some chargebacks are legitimate (stolen cards, disputes, fraud and the like), other chargebacks are the result of customer error and can be reversed by simply following the chargeback process prescribed by the card networks.

Similar to onboarding, to be effective the chargeback management API needs to automate the process fully, avoiding costly and time-consuming manual steps on the part of the PayFac. The API needs to not only allow tracking of chargeback activity by sub-merchant, but it needs to accommodate the variance of rules and policies associated each card brand. For example in the case of a dispute involving MasterCard, the decision of whether to go to arbitration is made by a merchant whereas with VISA, the decision resides with the card issuer. A chargeback management APIs should automate tracking and interactions through all phases including retrieval requests, chargeback initiation and pre-arbitration or arbitration. It should also provide programmatic interfaces for common tasks like assigning chargebacks to the correct sub-merchant, allowing sub-merchants to assume liability when appropriate, adding notes, and providing requested documentation via the direct upload of binary files to support a case.

With a proper chargeback management API, PayFac application designers can build chargeback management features directly into the custom interfaces that they present to their sub-merchants. They can better manage risk (by identifying frequent sources of chargebacks), avoid unnecessary charges, and maximize revenue and pofitability both for sub-merchants and themselves.

Merchant Funding

The whole point of a sub-merchant signing up to your service is to get paid, so doing a good job in this area is essential. Small PayFacs may be able to manage paying sub-merchants themselves, but at any scale, automation via APIs becomes essential in this area as well. PayFacs should have access to APIs that allow them to provide precise funding instructions, easily moving money to sub-merchant accounts, various holding accounts and to their own accounts as well.

By utilizing an API rather than web-based tools that providers might offer to track, fund and report on merchants, developers have the flexibility and control needed to devise creative new business models. For example they might levy particular fees for different types of services, or offer their sub-merchants incentives or compensation based on tiered revenue attainment structures. They might want to offer flexibility to sub-merchants like the ability to perform deposits across multiple bank accounts to make their offering more attractive to partners. PayFacs can also precisely control the schedule for funding, and can consciously withhold payments for contract or risk related reasons with the right APIs.

While a PayFac may not need all this flexibility out of the gate, as revenue grows and business models mature, flexible funding APIs help ensure that systems don't get in the way of delivering new capabilities that can provide a source of competitive advantage.

OmniChannel Transaction Support

The way that consumers make payments is changing fast, with mobile payments, digital wallets and other forms of stored credentials becoming increasingly important. A PayFac might start out accepting credit cards on a mobile-optimized web-site but this storefront might be quickly augmented with a downloadable mobile app, or even retail locations requiring card present solutions. Similarly, the types of payments accepted might expand.

The payment APIs provided for handling sub-merchant transactions need to be flexible and support a variety of card-present and card-not-present payment methods. The last thing a PayFac developer needs are separate sets of transactional APIs for each method of payment or mobile wallet the incorporate adding complexity to their design and complicating downstream maintenance. Ideally a customer who makes a purchase using one channel should be recognizable in future when they purchase via a separate channel.

Security & Fraud tools

Last but certainly not least are features related to security and fraud. While all developers need to design for security, this is an especially big issue for PayFacs. A single security incident can impact your brand and make all your sub-merchants skittish. Developers need proven APIs that can help them avoid storing or transmitting cardholder data to reduce PCI scope and ensure that they and their merchants are secure. As mobile wallets become more popular, developers would ideally like to pass mobile payment credentials directly to their processor, tagged to the appropriate sub-merchant and leave it to the payment processor to worry about details like decrypting tokens and vaulting payment credentials for future use. Rather than relying on fraud scoring mechanisms that are implemented globally and apply to all merchants, PayFac developers need the flexibility to tailor policies individually by sub-merchant, and even allow sub-merchants to tailor these policies themselves at the PayFac's discretion. This calls for granular API level features whereby developers can request additional "telemetry" with each transaction so that fraud-related metrics can be compared with acceptable thresholds configurable (within reason) on a per sub-merchant basis. By providing sub-merchants with these kinds of capabilities, not only are transactions made more secure, but integrations are simplified and PayFacs can provide value-added features in their software that help differentiate them from competitors.

Vantiv PayFac meets you on the road when building a payments platform. We continue to build, define and reinvent the rules of payment facilitation in partnership with the card brands. To learn about Vantiv's PayFac specific APIs and access our technical documentation, visit our developer community built for PayFacs at https://developer.vantiv.com/community/payfacs

vantiv®
smarter/faster/easier/payments™