Payment APIs: What, Why, and for Whom?

An Introduction to Payment Interfaces & the Kenyan Market

Date: 25 March 2016

259 Elm Street, Suite 200, Somerville, MA 02144 ▪ USA
Phone: + 617 628 0711 ▪ www.bankablefrontier.com

TABLE OF CONTENTS

“These [API] platforms are like the reefs or rainforests that provide life and substance to the other members of the habitat… Like a real ecosystem, these cyber-bio-systems will grow if their inhabitants/users can thrive. This means that API platforms must enable the success of their users, partners, and other constituents. This isn't an easy thing to do, but it's the opportunity that lies at the feet of anyone who recognizes what an API is and has the creativity to find ways to incentivize different stakeholders to join in.”

- Travis Spencer, Nordic APIs

Executive Summary

Open Application Programming Interfaces (APIs) are a necessary component of a robust digital ecosystem, and critical to the health of the surrounding economy. These interfaces allow businesses to connect in a manner that enables each to focus on developing and perfecting their core services, while relying on the others for areas outside their expertise. Conversely, a lack of open APIs results in a corresponding lack of efficiency, with each business implementing imperfect copies of the same business logic.

In this report we provide insights and guidance for those looking to learn more about User Interfaces (UIs) and Application Programming Interfaces (APIs), particularly those considering the use of APIs to provide digital financial services in the developing world.

A UI is the interface at which digital information is turned into a visual representation for *human* interaction, for example a mobile app home screen or a USSD menu. In contrast, an API is an interface for *computers* to communicate with each other, enabling collaboration across networks that each specialize in a given set of tasks. Paypal, Square, Stripe, and more recently, M-Pesa, are examples of services that expose APIs focused on digital payments.

We divide APIs into three categories: private, public, and open. Private APIs, as the name implies, are proprietary to a given company and intended solely for transmission of sensitive information across a closed network. Public APIs provide a limited set of functionality to a curated set of trusted partners through standardized communication channels called API endpoints. Open APIs are like public APIs but open to the world, employing interactive public documentation, an automated onboarding process, and even code samples and libraries. Providers of open APIs focus on providing a quality developer experience to those looking to integrate, and are subsequently rewarded with a much broader client base.

There is a natural progression for online services to evolve through these various API phases, with risks (e.g. security, reliability, resources) and benefits (e.g. scalability, growth, mass adoption) weighed at each step. In comparing the perspective of technologists in Kenya with our experience in more mature payments ecosystems, we have found that there is a relative lack of fully open APIs in the former. While there has been progress made in connecting digital services in Kenya, it is still the case that developers of new digital products there must make more concessions in their feature set and user experience than do their counterparts in regions with more open ecosystems.

Open APIs foster a more collaborative and robust business network where they are implemented. In regions like Nairobi, while M-Pesa is pushing forward through this evolutionary process as a Public API, this service and others have not yet evolved into open platforms. Developers are eagerly awaiting this transition, but the emergence of this platform-based economy is hampered by economic, technical, security, and competitive considerations around the scaling that must be outweighed by the benefits.

While M-Pesa has made progress in evolving from the “product” ecosystem of fixed UIs and private APIs, it has only just begun touching on the modular set of public APIs and embeddable UIs. What the developer community needs to support great products, wider spread innovation, and faster time to market is a set of fully open API features.

To this end, we also examine potential steps toward the creation of new opportunities for openness and interconnectedness among services. As a starting point, we can greatly encourage this development by providing a deeper understanding of the benefits of these somewhat technical concepts to the less technical pieces of the industry, while encouraging a better set of tools and documentation to the technical players in the form of a well-designed developer experience.

Open APIs allow an organization to focus on their core competency by collaborating with others who have perfected their own specializations. In this way, efficiency is increased overall and costs drop, in terms of allocated resources and/or time to market. This lowered cost translates to an increased margin for social impact and financial inclusion initiatives where costs or timing were previously a prohibitive factor. This is especially the case for resource-thin startups looking to use Open APIs. For them, any additional time provided to research and learn from their clients, identify potential pitfalls, and remove uncertainty inherent in their markets through innovation, can increase the likelihood of long-term success for the company.

Through openness, the financial technology (“FinTech”) space in particular will be afforded the ability to consider pursuing a whole new set of opportunities that were previously not possible. By building a symbiotic relationship with companies in the space via Open APIs, digital service providers have the potential to foster growth in this active ecosystem while also maintaining the pursuit of their own best interests.

Introduction to Interfaces

In order to provide a bit of context as to where an API fits into the grand scheme of things, we'll need to begin with a brief discussion of interface types. An interface generally describes a means by which information is translated from one form to another, or through which data is transmitted from one location to another.

A business looking to include FinTech services needs to understand which interface type suits their current and future product needs. Conversely, for a business that provides FinTech services, a crucial first step in planning growth is to identify which of these interfaces existing and potential consumers are looking for.

With these high-level goals in mind, we'll first proceed to describe what differentiates each specific type of interface, and subsequently take a deeper dive into the more technical details.

Key Takeaways

1. Interfaces, while typically associated with an interactive visual representation like a mobile device or a website, comprise more than an end user interface.
2. The wide array of interface types that exist are tailored to a specific set of parties involved at either end of the interface. User Interfaces translate digital information to humans, while APIs exchange data between machines.
3. The distinction between API types is typically around the level of access that is provided to the data, and how that access is administered.

User Interfaces

The interface that is most regularly presented by a technology company is the User Interface (UI), an interface that is targeted toward an end user of a given commercial product.

Product UI

A Product UI can generally be understood as the interactive visual piece of a product or service that a user physically interacts with. In other words, the UI is the point at which information is translated from a digital scheme into a human-digestible format. This includes everything from native mobile apps like Facebook and WhatsApp, to LCD screens and keypads on a POS machine, to a shortcode and numeric menu in a USSD application.

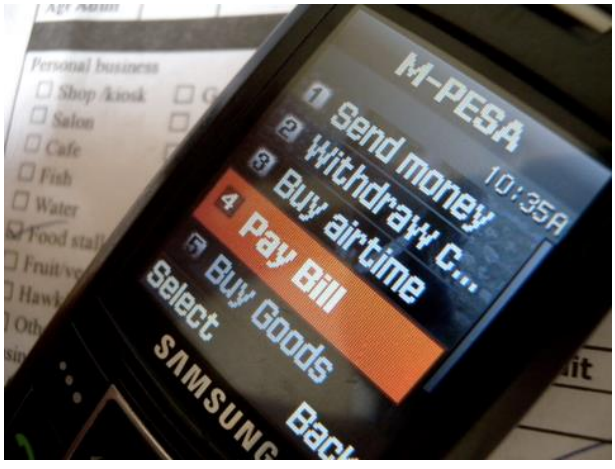

Figure 1: M-Pesa USSD Menu

Within the developing financial inclusion space, these specific descriptions map

readily to products already in the market. For instance, in Kenya, the M-Pesa SIM Toolkit user interface is available on nearly every phone, and many m-wallets including M-Pesa provide agents equipped with billpay systems, including USSD-based apps.

These interfaces each typically provide one boilerplate template that can be filled with content from a given provider. USSD shortcode applications, for instance, can be tailored to a business case, but have rigid restrictions on what characters can be included (alphanumeric only) and what structure the app can take (nested numeric menus).

Embeddable UIs

In addition to targeting end users, some of the more successfully adopted user interfaces have been extended to allow for inclusion in third party applications. In other words, the embeddable UI allows a third party business's development team to easily copy what is typically a few lines of code and paste it into their own mobile app or webpage to include a key feature from the experts' product.

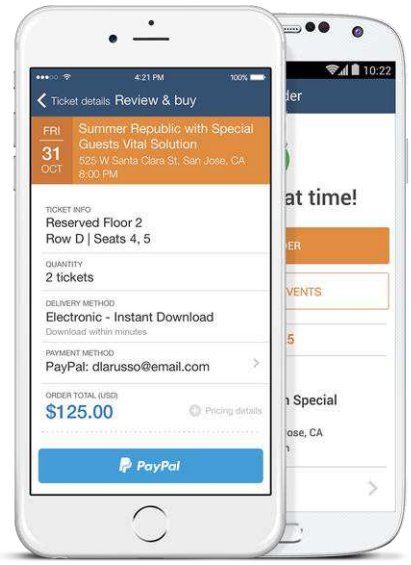

Figure 2: PayPal Specialized Interface

In the developed world, there are established providers of these types of interfaces. Paypal and Stripe are notable examples in the payments space that provide simple payment UIs that are incorporated into a wide variety of websites and mobile applications.

In developing markets, many electricity providers, television services, and other utility companies are now providing bill payment services via their websites. To facilitate this, payment processors that currently require custom integration for each partner are beginning to explore how to best provide pre-packaged UI services that can easily be included on third party sites.

While specialized UIs typically provide a bit of flexibility in terms of custom presentation to the user, even the more developed of these integrations are still typically limited to a few customizable parameters like the brand colors and logo of the company providing the service.

Application Programming Interfaces

As opposed to the User Interfaces (UI) discussed so far, Application Programming Interfaces (APIs) are designed primarily for machines talking to machines, only producing activity logs and reports for administrative human oversight.

The New York Times recently touted Application Programming Interfaces (APIs) as "the building blocks of the app economy… greasing the exchange of information [like] a modern-day Rosetta Stone." By designing a means to allow machines to electronically speak to each other in an ecosystem independent of application architecture, industry specialization, or

geographic location, many opportunities arise. We take a look at examples of some of these API types below.

Private APIs

A private API is an interface between a closed set of computers, typically designed for one specific purpose. By design, access to these networks is controlled in an extremely tight manner, so security considerations take precedence over scalability. These self-sufficient networks are typically maintained by an internal IT team or a consulting firm that makes updates to the network on a relatively infrequent basis.

Examples of the private API in the digital world include services like the channels fostering interbank transfers, advertising networks, private data synchronization, mobile network operators, and more. These are technically APIs in the sense that they are the interface over which programs to speak to each other, but their limited access makes them less relevant to the general public.

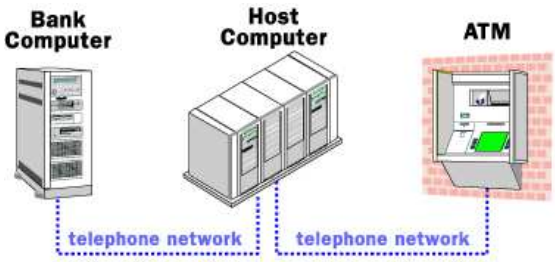

Figure 3: Simplified ATM Network

In spite of these services not being designed for public consumption, in our research we did see these private APIs being used as building blocks for public interfaces. For instance, before M-Pesa's API was made public, it was already working behind the scenes to power the aforementioned SIM toolkit and USSD application UIs.

Public APIs

Public APIs provide access to a specialized network of computers, but unlike private APIs they also allow access for a curated set of partner services outside of the API provider's private network. These interfaces are responsible for a subset of features and functionalities from the API provider's core expertise, and lean on other public and open APIs that are better suited for other tasks.

Opening up access while maintaining a narrow scope of services in this manner allows for a range of connected interfaces to be built on top of a public API, while maintaining as much of the privacy and security of the private APIs as possible. For UIs powered by a public API, in contrast to the constraints on customization imposed by a specialized UI, the public API allows for a decoupling of the data it provides from the visual interface built to present that data. Public APIs also typically provide public documentation published on the web.

Merchant service providers have provided public APIs for credit card based payments in many mature spaces, with access to these APIs controlled by an acquiring bank. For example, Chase Bank controls access to their Paymentech API and US Bank controls access to their Elavon API. In Kenya, merchant services are similarly provided via public APIs, with banks and mobile money providers determining which partner organizations get access. M-Pesa's newly public API is an example in that region.

Public API providers often select clients of their API manually via face-to-face meetings, email conversations, and other human interactions, with access only provided once the necessary business paperwork is signed.

Open APIs

An open API is a public API that has had the onboarding process streamlined, and is designed primarily around maximum scalability and providing the highest-quality experience for third-party developers tasked with the integration of services.

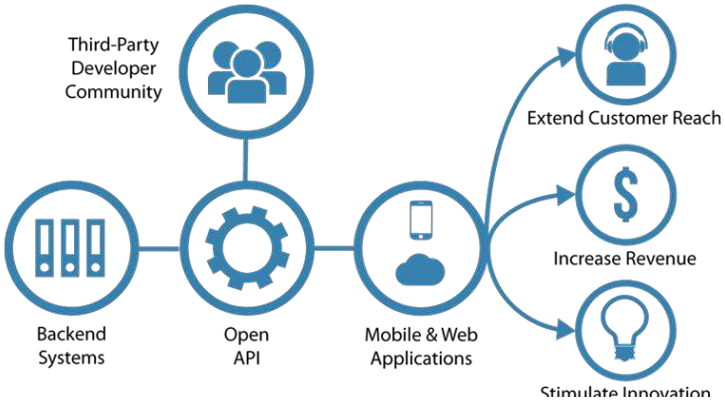

Figure 4: Simplified Open API Ecosystem

Rather than requiring a manual vetting process prior to allowing access to new partners, open APIs provide an automated process for gaining access to these services, libraries for integration, open sandbox environments for testing them, and an autonomous launch process.

Stripe and Paypal provide open APIs in addition to their specialized interfaces in order to allow for non-standard payment process flows and UIs. PesaPal is a similar provider of an open payments API in Kenya, but does not include the p2p features.

Platforms

The terms “platform” and “open API” are often used interchangeably, but some make a subtle distinction. While an open API necessarily provides an interface for computers to access a service, a platform is distinct in promoting this interface as the *primary means* of interacting with the service. Nordic APIs, an API-centric event organizer and digital publisher, provides a more thorough list of criteria for what a platform, but not necessarily all open APIs, must provide:

- Enables access to the organization’s core value proposition
- Is technically and non-technically scalable
- Enables consumers to create shared value
- Is instrumental in securing the organization’s position as a market leader
- Is seen by top management of the platform as business critical

Throughout this report, we include platforms as a subset of what we refer to as open APIs, focusing on their common benefits. Illustrating this more vividly, Travis Spencer of Nordic APIs draws the analogy that’s quoted in the opening page of this report: open APIs (and therefore platforms) are like nature’s biomes in the sense that they support an ecosystem that thrives precisely because of the API’s existence.

Evolution of a Connected Business

Armed with a working set of nomenclature for interface types, we will now touch on how a digital service typically evolves over time, and what steps and considerations a business typically takes prior to opening up an API.

Key Takeaways

1. Businesses that enter the digital space tend to evolve in a relatively predictable way, with opportunities and risks to evaluate at each transition.
2. It is important that companies treat these evaluation points carefully and consider the implications of providing a digital service in the context of their product roadmap and growth strategy.

Evolutionary Phases

Core Business: The Product UI. A business typically begins its online life focused on a single service, which is provided directly to the end users. To launch this service, the business starts by building a UI that enables a specific set of use cases, perhaps even utilizing their private API services and external public and open APIs to accomplish this.

Reusable, Interconnected Modules: Public APIs and Embeddable UIs. If that initial foray proves successful and there is sufficient adoption of the services, the next step is often to evolve the service into a repeatable specialized interface. These take the form of the embedded UI and public API described above. The interface can be utilized internally, by third parties, or even licensed by competitors in order to minimize development costs.

Features as a Service: Open APIs and Platforms. Finally, if this interface becomes popular, the business may decide to expose services directly via open APIs to be integrated with other businesses, and in this manner reach a wider audience by choosing to become a platform.

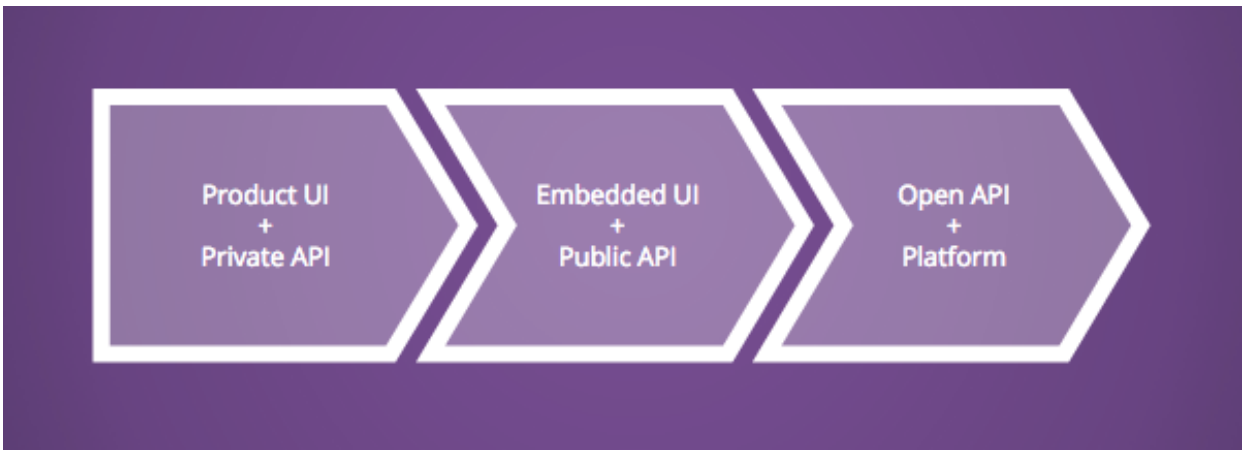

Figure 5: Evolutionary Phases

Opportunities vs. Risks

The evolutionary steps described above are common for an API provider, and the choice to grow within this framework boils down to the strategic rationale of considering the balance of opportunity and risks at each step.

Examples of such opportunities include:

- **Revenue Source:** Will the transition create new revenue opportunities by offering API access to other businesses and startups for a fee?
- **Efficiency:** Will opening up create greater efficiency for operations and for our existing customers? (e.g. automating processes, reduced time to market, more innovation, greater scope for customization, etc)
- **Market Share:** Is there a set of potential customers who aren't using our product but will utilize our service as an API?
- **Virality:** Will opening up allow our current customers to become more effective marketers for us, or to create services that magnify our growth velocity?
- **Corporate Image:** Can we promote this API as a competitive advantage in our marketing materials? Will we be seen as thought/action leaders in the industry?

The risks to be considered may consist of:

- **Initial Cost:** What are the costs of code development, infrastructure setup, and customer support associated with this transition?
- **Security:** What measures must be taken to expose services to the public Internet while maintaining a secure and reliable service?
- **Reliability:** What will it take to ensure this system can scale without outages?
- **Legal:** Are there regulatory and compliance considerations in opening up the service?
- **Reduction of Control:** How will you manage a reduced ability to neatly control what your service does, how it's used, who uses it? How will you manage a potential transition to volume rather than margin play?

Ultimately, it boils down to a provider looking at reach and revenue versus cost and risk. If the team determines that these benefits outweigh any properly mitigated risks, the company is ready to grow, and *should* grow, to the next stage.

Comparative Analysis: Payment APIs in the Kenyan Market

At this point, armed with enough of a working knowledge of types and phases of APIs, we're ready to take a look at the state of the payment API space in the Kenyan market versus the state of the industry globally. We will look at this from the perspective of some archetypical technology businesses, similar to our interviewees, who are looking to consume API services.

Key Takeaways

1. As an API Consumer, having a strategy for evaluating API Providers against your needs is critical. We suggest several high-level components for this evaluation: features and benefits, infrastructure and technology, developer experience, and business model.
2. Each step should help narrow down the available options, but ultimately the decision is likely to come down to an evaluation of the top candidates against business needs.
3. Continuing this evaluation as the market evolves is a critical step. Solutions that you previously "hacked together" because of technological constraints may be made more elegant or robust by a new feature or service that's been introduced since the last evaluation.
4. While M-Pesa has significantly progressed in their evolution from a semi-private IPN to a public API, they still lack some of the benefits of the more mature open APIs.

Step 1: Users of APIs

The types of businesses in the Kenyan space looking to utilize payment APIs include: m-commerce, mobile money distribution, mobile apps, subscription services, and money management applications, which are each described here.

The M-Commerce Site

To accommodate the pervasiveness of mobile devices relative to that of personal computers, many businesses have turned to M-Commerce solutions and mobile-optimized websites. One critical task for developers of these sites is to implement a method to charge customers for the items in their shopping carts, which is typically accomplished through integration with a digital payment processor.

The Mobile Money Aggregator/Distributor

Next are modern crowd-funders and lenders who receive payments from and disburse to mobile wallets. For example, developers of services like Inventure's Mkopo Rahisi app need a way to quickly perform actions on mobile money accounts like authorizing third party access, crediting value, and debiting them. Groups like the World Food Program also face these same requirements in distributing electronic cash transfers (or "bamba chakula") to refugees as a safer alternative to supplying physical goods like cash or food rations.

The Mobile App Provider

This case includes examples like an app store seeking to implement purchases for our stores in Kenya, or a smartphone app developer wanting to allow in-app purchases with a portion of the funds going to the app store automatically. We would be interested in exploring what options are available to us to enable these payments.

The Subscription Service

Subscription services for digital media, entertainment, and physical goods are at an all-time high and growing rapidly. We are in an era where video games like League of Legends can gross more than one billion USD in micropayments over the course of a few months, and postnatal products for infants and young families can be automatically paid for and delivered at critical developmental moments. A Kenyan developer looking to capture a piece of this market would benefit from the existence of a mobile money subscription service.

The Money Management App

Lastly, we'll view the landscape from the perspective of a Micro Finance Institution (MFI) that is planning to go digital-only and offer a smart money management app that suggests a tailored set of financial products to an individual based on their financial history. For a developer building such a product, the value of the product's insights to the end user are directly dependent on what financial data can be fed in. In other words, the more APIs that this app can connect to, the more valuable the insights that are fed out will be, and the more attractive and polished the product will be for the end user.

Because I didn't have an API…

While conducting interviews in Nairobi, Kenya, we discovered the following scenarios:

- I had to plan my business cases around potential dropped and delayed transactions
- We had to hire a dedicated team to manually upload user data every night
- There was a two-month delay and multiple meetings involved in creating a new business partnership
- Hacks and less elegant solutions had to be used, such as scraping and redirecting data from nightly uploads to an internal file system, that then has to be emailed to the financial audit and reconciliation team

If I had access to an API…

During the same interviews, we also got the following feedback on how APIs could help:

- My support team would experience greater efficiency. I could reassess where they spend their efforts, and potentially spend more time on research and development.
- We wouldn't have to manually parse emails to answer our partners' requests
- Reports and data analysis could be automated, and allow for better insights into what sectors of the market we should focus on for future growth
- We could better compete with the larger players who currently rely on this operational hurdle as a competitive advantage.

Step 2: API Providers

By researching the services available for these API consumers in Kenya, we gained some insights into what a few companies' offerings look like. Namely, we explored M-Pesa, PayPal, Stripe, Square, and Android Pay.

M-Pesa

In Kenya, Safaricom's M-Pesa product has recently undergone a key transition in the evolutionary process. After starting as a wallet-to-wallet payment service with a mobile UI, they evolved an additional private API for select partners to process bulk payments and other services, and most recently determined that it made sense for them to make their formerly private API endpoints public. They have indicated that they are in the process of beginning to evaluate the automation of their onboarding process in an effort to reduce friction, increase adoption, and eventually become a fully open API.

Paypal

Paypal, one of the pioneers in mobile and web-based digital payments, offers a robust set of Product UIs (mobile and web apps), Embeddable UIs (checkout button widgets), and open APIs for payments and reporting. They've more recently acquired Braintree, a purely API-based payment processor, and Venmo, a company focused on socializing P2P payments.

Within Kenya, driven by the launch of Google Wallet, Paypal started allowing users to deposit and transfer funds using their service. Within a year, they began to allow withdrawals to a US bank account, but relatively few merchants had this at the time. Finally, in late 2013, through a groundbreaking partnership with Equity Bank, they launched a withdrawal service that allowed Kenyans to utilize the Paypal UIs for the full set of features.

Stripe

In the US, Stripe made the decision to eschew the UI phase entirely and immediately provide an open API, as it provided a distinct competitive advantage. Their focus on providing a great developer experience for integrators above all allowed them to rapidly grew their market share relative to existing services.

Until very recently, the downside was that their services were geographically limited to a few regions, which excluded Kenya. Just weeks ago though in early 2016, sensing an opportunity in regions lacking an Open API for payments, they launched Stripe Atlas. This is a service through which entrepreneurs around the world can easily incorporate a U.S. company, set up a U.S. bank account, and start accepting payments with Stripe in regions where this previously was not possible.

Square

Square started with card-reader peripherals for mobile devices and a Product UI in the form of a mobile app that together made it easy for any small merchant to accept credit card payments using their phone or tablet as a Point Of Sale (POS). They also created Square Cash, a set of mobile and web UIs focused around P2P payments. Square Cash was first to allow transfers without requiring user registration. They provide free transfers by making interest

off of the funds while in transit. After success with each of those products, they chose to open up the payment and reporting functionality, introducing Square Connect as their Open API for all of these interfaces.

Android Pay and Google Wallet

Android Pay is Google's mobile-centric UI, built on top of the Google Wallet infrastructure, that allows for contactless payments and P2P transfers from a mobile device. Android Pay provides a Software Development Kit (SDK) for mobile developers looking to integrate with their service. Google Wallet provides an open API for payments and reporting also allows users to transfer funds on through product UIs and embeddable UIs. Gmail uses this feature of Wallet to allow "attaching" of payments to emails as a feature. However, Wallet's API does not support automated P2P transactions.

In the Kenyan market, Google initially partnered with Equity Bank in 2012 to pilot the use of BebaPay, a nfc-enabled card, for cashless payments for matatu (bus) fares. Customers could top up their card at Equity Bank agents or by using Google's services on an NFC-enabled Android phone. However, even with 100k users, the service was deemed unsuccessful and in 2015. It was suggested that customers instead transition to Equity prepaid MasterCard for these services, which can only be topped up via mobile money or a bank agent.

Step 3: The Criteria

In order for an API Consumer to make a decision on which of these API Providers to use, they must evaluate the quality of each API's features, infrastructure, development experience, and the business model. In this section we discuss each of these criteria in detail.

Arguably the largest development in the Kenyan financial API space in the recent past is the previously mentioned evolution of the M-Pesa Private API into their Public (but not Open) API. Thus, for each of the criteria that follow, we evaluate in the context of the previously presented API consumer archetypes the following three scenarios:

- The Recent Past: What could I do before the M-Pesa APIs?
- The Tight Present: How has that changed now that we have some APIs?
- The Open Future: What would it look like if a fully open API (e.g. Stripe, Paypal) were available directly in Kenya?

A summary matrix is contained in **Appendix A,** which may be useful to reference throughout the more in-depth analysis that follows.

Features and Benefits: Usefulness of an API

Business to Business (B2B)

Prior to the M-Pesa API, any of our archetypes that were looking to facilitate the exchanges of funds or information between businesses would find that just two of our six API providers offer services targeted specifically for that case.

Paypal's Mass Pay API and Adaptive Payments products allow a business to define custom flows and triggers for their B2B transactions, their Next Day Settlements allow for movement of money, and group invoicing products allow for up to 100 simultaneous invoices. **Stripe** Connect is a platform used by B2B companies like Kickstarter and Shopify to send payouts to partners, with discounts available at scale. In addition to the APIs for exchange of funds, each of these services provides APIs for pulling transaction history and other information.

M-Pesa historically did not offer explicit B2B services, although businesses did use the wallet UI for this purpose. In an effort to ease friction in cross-border payments in East Africa, in one key solution to onboarding businesses to its service Safaricom forged a partnership with MTN beyond Kenya's borders to allow for cross-wallet payments internationally. Shortly after this launch they released a report on how that onboarding has translated into usage, in which they reported a 65% increase in manually entered B2B payments within Kenya.

They now offer automation of many types of fund transfers in their new M-Pesa "G2" API. However, compared to the other players, they are notably lacking the automated invoicing and customization of payment flows that PayPal offers, and a clearly defined volume discount that Connect promotes.

M-Pesa also lacks a fully automated way to transfer information such as a business or consumer's transaction history. MLedger was given private access to M-Pesa histories after Safaricom acquired their parent firm, Dynamic Data Systems, in 2014, and includes a UI feature that allows users to export transaction data manually. While a "Transaction Extract Specification" for this feature was included in M-Pesa's public API documentation, it is technically not an API endpoint that allows for real-time data retrieval. Instead, a developer must trigger the generation of a report, followed by some delay while the file is generated, after which the file can finally be retrieved over secure file transfer protocol (SFTP).

If a full payment history API were implemented, the Money Management App discussed earlier could implement more efficient features like pagination, searching, filtering, and more by sending parameters to an API. Instead, the App is expected to create a complete copy of the data on their own servers and build their own private API implementation layer over it, both of which could be prohibitively costly for an organization in its infancy.

Moreover, financial inclusion is dependent on increasing the veracity of this financial data to the point that it's usable for financial product suggestions, which in turn is fully dependent on reducing the number of steps between collection and use. Providers like M-Pesa must open historical transaction API endpoints for this to be possible.

Business to Customer (B2C)

If a business such as the Mobile Money Distributor were looking to consume a business to customer payments API (e.g. disbursements, refunds, payroll services), they may take note of the same two players as the B2B market.

Paypal's Adaptive Payments defines Actors and Objects in a "one-to-many" service that map readily to businesses and customers. Paypal's subsidiary **Venmo** also offered a beta Payouts service, exposing API endpoints for disbursements to Venmo users, but the service has now been discontinued in favor of Paypal's. **Stripe**'s Transfers API allows a business to "pay arbitrary third parties using your Stripe account," but has been deprecated in favor of their Managed Accounts, which allow for better identity verification and more international support.

In Kenya, prior to **M-Pesa**'s public API, services like our Mobile Money Distributor would have had a hard time coming into existence. They would have needed a team of agents to manually enter transactions into company-controlled mobile wallet via a Bulk Disbursement web interface, which supports cases like loan disbursement, insurance claims settlement, and daily wage payments are all supported through this UI.

M-Pesa has made this service public via their API as "Automated Payment Disbursements" to better support larger and automated disbursements. However, this is only possible after the developer completes a well defined, but somewhat lengthy onboarding process.

This welcomed move allows B2C businesses to employ machines to take care of this menial task, and make better use of the human talent previously required to interact with the UI. Opening this up through an automated onboarding process will ease this friction even more.

Customer to Business (C2B)

For the C2B-oriented M-Commerce Site and Mobile App Provider, the API provider space is a bit more crowded, and we have all of our candidates providing services here. They fall into two broad categories: open APIs backing embeddable UIs, and product UIs on customized hardware.

Paypal provides custom C2B payment flows, laid out in an introduction to their API. For quick integration, their embeddable Express Checkout button can be included in a site relatively easily. The button redirects customers to Paypal's product UI, then back to the original UI for confirmation. **Venmo** provides a Payments API for deeper integration of their services for C2B payments and a tool called Payment Links that generates an embeddable link for a static amount, which can be included on a third party web or mobile service. **Stripe** also provides an introduction to their Connect API for custom payment flows, with the embeddable Stripe Checkout UI element that processes the payment without redirection.

Square tends toward the product UI route, offering their Reader, POS, and Stand hardware with pre-installed software, requiring no technical knowledge in order to start processing payments in a physical retail shop. For customized solutions, they provide an introduction to their Connect API, which is more relevant to our M-Commerce and Mobile App Provider. Android Pay and its API allow for secure payment capabilities to be integrated seamlessly into any Android UI for in-app digital payments or credit card emulation for brick-and-mortar retail.

M-Pesa provides C2B payment UIs for customers who top up their wallet via an agent or Bank to M-Pesa USSD services. Until recently, C2B payments have been handled via the semi-private Instant Payment Notification (IPN) service. In order to use this, a business first needs to file for a paybill number and set up an online server to receive notifications when payments are received. Like M-Pesa's B2B and B2C services, this has also been modernized and opened up via the M-Pesa API as "Automated Payment Receipt Processing."

While these developers found IPN services sufficient for a number of merchants to receive and store transaction information, the notifications were sent via one-way channels like SMS that didn't allow the parties involved to take any other subsequent actions when needed.

With the introduction of M-Pesa's public API, developers saw this previously one-way unconfirmed payment system evolve to a more interactive service, and this benefit could be passed to end users of a service. Developers can now present confirmation dialogs to ensure the customer has entered the correct M-Pesa account number, and can automate refunds and reversals. These features significantly reduce chargebacks, which in turn reduces both interchange rates and the customer support team needed to handle these reversals.

At present, there is a lack of any open API or library that allows for straightforward inclusion of mobile money payments directly into mobile apps or stores. The industry needs a service that provides a drop-in SDK library so developers can move past this challenge and start collecting mobile money payments with minimal effort.

Moving forward, M-Commerce and Mobile Application developers need even deeper integration and more automation. They would greatly benefit from a Stripe-like service reducing the signup process to a 60-second web form, and simplifying integration literally to several lines of code.

Bill Pay and Invoices

Our Subscription Service provider would look to examine three of our six API providers, which have products specifically tailored toward automated billing and invoice payments functionality.

As part of **Stripe**'s core API, their Subscriptions service automates subscription plan creation, customer signup, subscription upgrades, renewals, and more. **Paypal**'s invoicing services include a web-based UI, a widget for integrating directly into excel spreadsheets, and an API to automate this process. Similarly, **Square** Invoices can be sent out via a web UI or integrated via their API.

In addition to it's primary use of providing a B2C "buy goods" feature, **M-Pesa**'s Lipa Na M-Pesa UI offers a billpay service that's used for utility bills, school fees, prepaid card topups, rent payments, and more. Their API's "Automated Payment Receipt Processing" can be used

if additional validations or automation are required. However, neither the historically private M-Pesa API nor the newer public API includes a subscription feature to collect monthly fees.

These Subscription Services of the digital commerce space could be fostered and accelerated in Kenya by introducing endpoints into the existing APIs, or by the creation of a middleware subscription API solution for developers that queues each one-time subscription payment and calls the limited existing APIs at each payment due date. Until then, we will be limited to the most driven developers being bogged down building their own subscription modules prior to launch.

Person-to-Person (P2P)

For a business like the Mobile App Provider who may be looking to allow P2P payments feature, there are limited options available.

Paypal's Adaptive Payments (discussed previously in the B2B section), which **Venmo**'s API has been deprecated in favor of, allows developers to build custom embeddable UIs or to automate payments via the API. Alternatively, **Square** Transfers API allows for automated transfer of funds from one user to another via ACH, using the merchant's account as an intermediary. This means fine-grained control over what, when, and where money is sent to/from a Stripe account. However, this also means that the merchant needs to keep track of how much is owed to each of the users, and trigger payouts explicitly.

M-Pesa has a robust and widely adopted set of P2P UIs in the form of USSD menus and SIM-based applets. However, their API does not include any explicit P2P features. This is unfortunate since according to their half-year report in 2015 this is their most popular feature, with P2P accounting for over 75% of M-Pesa payments.

By opening up P2P services via an open API or even a public one, M-Pesa would compete more readily with Paypal and Square, allow for tailoring of P2P UIs for specific industries, and capture or retain portions of the market that need this capability.

Marketplace

A couple of these services offer products that facilitate the process of setting up and running an online marketplace.

Paypal provides the Braintree Marketplace, simplifying the splitting of payments between providers and the marketplace administrators. Several aspects of this can be automated via their API: onboarding sub-merchants, confirming the sub-merchant onboarding, creating transactions with service fees and holding transactions in escrow. **Stripe** provides the previously mentioned Stripe Connect API that is not explicitly for a marketplace, but has features that could be used to build a marketplace on top of in the manner that Lyft and Kickstarter have done.

Like Stripe Connect, **M-Pesa** doesn't provide a direct solution for marketplaces. So in a similar manner the merchant-centric portions of their API mentioned above could also be used for

this purpose for businesses like our M-Commerce site that may look to evolve into an online marketplace.

Financing

Many of the services provide financing options that typically rely on a business's history within the API Provider's service to assess eligibility, and give credit based on that.

Paypal's Working Capital uses a small business's history to assess what level of credit they can borrow against, a process which Paypal says can be completed "in minutes." **Square** has a similar service called Square Capital, that is distinct in that the repayments are tied directly into the card sales, so that a merchant pays more off when business is strong, and less if things slow down. **Stripe** partnered with a lender called Kabbage to enable businesses that process payments online to apply for financing. Businesses that haven't recorded payment revenue before won't be able to apply for financing. **Venmo** leverages a similar partnership with Ledge to offer lending services, using its strong P2P network to foster financing via personal crowdfunding rather than being calculated from the business history.

While **M-Pesa** has consumer financing services like M-Shwari built on top of it, there is no direct financing for partners built into the API. This is a critical feature for a startup in developing parts of the world in particular, which may not have access to capital like venture capitalists or bank loans. By building this service in, they have the power to foster a healthier and more diverse ecosystem of startups that can reach deeper and more broadly into the market, fostering more financial inclusion.

Infrastructure and Tech: Evaluating Robustness

After reviewing features, the next step for all of our API Consumers is to begin researching tech requirements for each API Provider. This includes security and authentication mechanisms, fraud protection, data extraction and reporting, and availability.

Security & Authentication

Regardless of the industry, interfaces must secure data they expose via an API or UI to the Internet. This is measured by the encryption mechanism used to protect the data in transit, and the authentication and permissions protecting the data at rest.

For developers, OAuth2 (an authentication standard that allows the end user of a service to control which of their data is exposed to connected services) and TLS (encryption standard for securing any transmitted data) are the most straightforward combination to implement. **Stripe**, **Square**, and **Venmo** each offer integration using these standards. **Paypal** continues to offer OAuth authentication, the more tedious predecessor to OAuth2. Even when using libraries that abstract the implementation details away, there is still added effort with this standard, with no benefit over the more modern counterpart. **Android Pay** provisions static API-level credentials. Compared to OAuth2, this is less secure, since a breach that leads to these credentials becoming public will compromise *all* users' security as opposed to just one.

M-Pesa requires the combination of static API-level credentials and a Virtual Private Network (VPN) to connect to their service. While this combination is more secure than just API-level credentials, it's not a quick and painless integration. Imagining private data as water, TLS is a secure pipe that the water can flow through without anyone seeing what's inside. A VPN can essentially be used as a reinforcing layer over that pipe that takes extra time and effort to get set up when connecting services, but takes effort to implement.

If M-Pesa were to offer an interface with OAuth2 authentication and TLS encryption, they'd not only at least maintain the security level they have in place, but would likely see more rapid growth due to the reduced effort for integration.

Fraud Protection

In order to build a service around digital payments, fraud reduction and mitigation must be a key part of the analysis. Thus, it is a positive mark in evaluation if a public API or platform has guidelines around this.

Stripe provides fraud prevention guidelines and features in a thorough article on the topic within their support site. **Paypal** also has comprehensive fraud protection guidance and configuration, which is laid out in their developer documentation. **Square** has a section of their website dedicated to security that discusses the measures they've taken to secure their services and security tips to integrate with them. **Venmo** discusses their security practices on their site without explicitly laying out fraud protection strategies. **Android Pay** has a section of their documentation on security practices, including Masked Wallet Requests, which are Javas object containing masked details. Comparing each service, Paypal and Stripe are clear leaders in this area.

M-Pesa should follow suit by offering a set of best practices. This will inevitably reduce the incidence of fraud, increasing their margins that they could choose to pass on as lower rates.

Data & Reporting

When running a service that's integrated with payments, disbursements, and the like, access to the data and tools to interpret it are critical. Reconciliation and anomaly detection can help a company ensure they are collecting the funds they are supposed to be, and data viewed in the right way can even help plan out next steps for the product.

Stripe listed their reporting capabilities in a support topic; monthly summary, balance history, account data, stripe connect partners data are all included in the web dashboard. There are also modules to plug into QuickBooks, Xero, or NetSuite directly via their API. **Paypal** has extensive documentation on their reporting capabilities, including search capabilities, monthly sales activity, order fulfillment, and customer agreement details. This information is available via UIs like Paypal Manager and also exposed via APIs like their Payflow Gateway and Merchant API. **Square** provides a web-based dashboard for analytics on their POS services as well as anything processed via the API endpoints including: sales summary, sales trends, payment methods, item sales, category sales, discounts, modifier sales, taxes, transaction status, gift cards, transactions, deposits, and cash drawer sessions.

Android Pay and **Venmo** are much more limited in their reporting features. In terms of providing a user-friendly interface with a rich dataset, Square seems to be the current leader, with Stripe and Paypal close behind.

Unfortunately, M-Pesa does not discuss reporting capabilities or data in their marketing or developer materials. As discussed in the B2B discussion above, this is a missing critical feature for businesses to implement metrics and machine learning models as product features.

High Availability

When providing mission-critical functionality like payment processing and other financial mechanisms, availability is key. Many API services will tout "four nines" reliability (99.99% uptime), which translates to less than an hour/year of interrupted service. Others will lay out explicit Service Level Agreements (SLAs) on a partner-by-partner basis, depending on their set of needs. Unfortunately, none of the services we investigate here provide uptime data or SLA information publicly at this time, so there is no clear leader here.

If M-Pesa were to guarantee uptime, they would immediately have an advantage over the competition as well as reassuring API Consumers that customers can count on their service.

Developer Experience: APIs as Building Blocks

Often overlooked, the developer experience is a critical portion of this process that should be carefully evaluated. By understanding the value a platform places on external developers, one can gain insights into how long an integration may take, how responsive the service's integration team may be with any questions during onboarding, and what to expect for support and maintenance after launch. Minimizing friction in each of these processes means less distractions and wasted time for a development team, which ultimately translates to money saved on integration and recurring overhead.

The concept of Developer Experience (DX) is a relatively new, but rapidly growing concept. Sprung from the principles of the broader User Experience (UX) domain, the idea is that a happy developer is an effective developer. Beyond that, with businesses interacting more and more on the digital playing field, developers' satisfaction is working toward claiming its place as a key factor in business development.

Jeremiah Lee Cohick of Fitbit recently summarized DX as having four main components:

- **Functionality**: what problem the API solves and how good it is at that. An API should not only strive for effectively solving a problem, it should also solve it well.
- **Reliability**: a set of non-functional requirements like availability, scalability, stability, that build trust and represent a necessary ingredient in driving developers to use an API.
- **Usability**: how well does an API lend itself to discover and learn how its functionality can be used ("intuitability")? How easily does it allow developers to create tests? What support does it provide for error handling?
- **Pleasure**: how enjoyable is an API to use?

Amit Jotwani of Intel Mashery put these general ideals to more practical terms, which each translate well into concrete metrics that can help decide how well a technical partner's integration will be received by a developer:

- Keep it simple.
- Avoid marketing jargon; lay out what developers can stop doing if they use your API.
- Simple, fast signup, which includes easy management of API keys, service selection, etc.
- Fast setup: ideally, only 5 minutes should be required to create an "Hello World" app.
- Fast API key provisioning to spare developers long waits.
- Clarity about costs/limitations; don't create excessive expectations; provide free trial.
- Provide stellar documentation that is complete and up to date, and avoid using PDFs.
- Provide interactive documentation to make it easier to discover and learn APIs.
- Show code, providing clear, concise examples of how an API can be used for a given task.
- Inspire developers and be available on channels such as Stack Overflow, Twitter, etc.

The results of an evaluation against these key criteria provide a great indicator of whether an API provider is intimately familiar with the service they're providing, and whether they have consider their relationship with the end user of their API as valuable. DX is at least as important as any technical criteria that follow.

Of the Kenyan perspectives previously enumerated, the M-Commerce Site and Mobile App Provider in particular are examples of companies that are seeking better DX in the form of better documentation, more automation in the onboarding/signup process, code samples and libraries.

Key points to look at in this process for any API include: how well the documentation is composed, whether an online forum or support channel exists, whether there are libraries or SDKs that are already written that can more-or less be just "dropped in," what the onboarding process is like and if it is automatic, what degree of control a developer has over the integration plan, and if the company engages with developers outside of normal business in hackathons and such. Happy developers mean good code and a better business. Making sure the developer experience is great is at least as important as any other topic here.

Key Takeaways

1. In the digital domain, developers are the people building inter-business relationships.
2. To ensure these relationships are sustainable, it is critical that developers are recognized as a key component of the growth and success of a company looking to thrive in the digital space.
3. Concepts like Developer Experience (DX) and other technical criteria provide a great set of rules to evaluate prior to integration.

Interfaces and Endpoints

These are the primary consideration from the development perspective, as they define the basis of how the two systems will communicate. The specific manner in which a developer is allowed to set up their software to talk to pieces of the API interface (known as "endpoints") is a critical part of the vetting process.

For the API Providers we're looking at, the first consideration is a choice between Representative State Transfer (ReST) protocol and Simple Object Access Protocol (SOAP), examples of which are illustrated in **Appendix D**. In the absence of an explicit reason to use SOAP, the rule of thumb here is to use ReST, as it does not require a tight coupling between the API provider and consumer so each can independently update code without "breaking" the integration. SOAP's tight coupling, on the other hand, is useful when a strict schema is necessary. For native web and mobile-based applications, ReST nearly always wins out.

We see here that **M-Pesa** and **Android Pay** are the only two API Providers that do not offer a ReST interface. The former has only recently launched their services as a public API, and has thus far been able to rely on the coupled nature of SOAP to provide a strict format for interaction with a controlled set of API consumers. The latter service is mobile-centric and relies on Google Wallet's API as a ReST option.

Webhooks

A "webhook" is a notification to a third-party server triggered by some event in an API service. This is important in payment processing, as batches are often only settled on a daily basis, and many other actions occur asynchronously. When API Consumers receive callbacks via a webhook, they can log the data for later use and even trigger additional actions like emailing a receipt, or setting up an item for shipping. **Stripe, Paypal, Square,** and **Venmo** all provide some form of webhook capabilities, and the others should look to follow suit.

Technology Stack and Protocols

In software, there are various parts to an API that can be collectively referred to as the "stack." These include what sorts of databases are used, what language the business logic is written in, and how it is presented to the end user. This also includes any libraries or software development kits (SDKs) that are provided to ease integration for third parties, like your developers. Typically the main consideration in this part is what format or "language" the information is returned in, and via which protocol. Examples of protocols are illustrated in **Appendix D**.

Documentation

The most important part of integration is easy-to-access, thorough documentation. Without this, a developer will not even be able to get out of the gate. Each of our candidates has documentation available in various formats.

Stripe has an extensive, interactive, online set of documentation with full descriptions, request parameters, response parameters, and even sample implementations of API calls in various formats and languages (cURL, Ruby, Python, PHP, Java, Node, Go) to get developers

started in an extremely efficient manner. **Paypal** also provides a high-level set of descriptions on their website for their multitude of products that are categorized into more detailed API documents including samples and descriptions of each call, along with useful sample implementations (cURL, Ruby, Python, PHP, Node, Java, C#). **Square** provides a robust set of online documentation, with clear descriptions of parameters, as well as some sample JSON request and response bodies, but no sample implementations for the developers. **Venmo** lays out high-level descriptions of their services in a set of online documentation, but only includes sample requests and responses where necessary. **Android Pay**'s documentation contains some high-level process flows for their unique payment structure functions, as well as suggestions for integration. They provide a tutorial for integration for a native Android app that includes some code samples to drop into an app.

Of the documentation, **Stripe**, **Paypal,** and **Android Pay** seem to provide the most extensive and easily accessed documentation, and include code samples to make integration extremely straightforward for developers.

Safaricom's **M-Pesa** API static documentation is available for download via their website in pdf and word format. It outlines each of the features, and includes some sample API calls and the SOAP structure for a request. However, there are improvements to be made: the documentation is not interactive, nor does it include code samples or libraries, which would greatly ease integration effort.

Online Forum

Once a developer has the documentation and begins integration, it's likely that they may run into an issue. For this reason, it's helpful to have a community of other people who are doing the same thing to bounce ideas off of. Many times, the companies providing the service will create an official channel for that type of communication.

Stripe has an active Google Group as well as an Internet Relay Chat (IRC) channel called #stripe for real-time communication with some of the core developers of the product itself. **Paypal** has an official "paypal" tag on StackOverflow for addressing developer questions. **Android Pay** leans on a Google+ community called Android Developers for online communication between developers of all Android solutions. There are also unofficial channels like email lists, but those are out of scope for this discussion. The clear leader here in terms of providing official channels to have questions and other feedback addressed easily and without any dilution across unofficial networks is **Stripe**.

While **M-Pesa** has held in-person forums for developers, there is no official online community for developers involved in integration. Creating and fostering such a portal would allow developers to not repeat the mistakes and inefficiencies of those who came before them, and would reduce typical integration time significantly.

Support

In addition to community members helping each other, each of these services employs dedicated support staff.

Stripe, **Paypal**, **Square**, and **Venmo** all provide online support mechanisms where chats can be held in real time and issues can be addressed in a modern manner. **Android Pay** relies on the same Google+ community that they use as an online forum.

M-Pesa support is limited to email and telephone services. This is a great first step, but does not necessarily scale well, and can ultimately lead to developers waiting on the phone when they could instead be researching answers in a forum, or programming another feature.

Unfortunately, none of these services publicly list premium support as an option, which critical for companies with large user bases and high uptime requirements. The lack of premium support services has the potential to hinder scalability.

Self-Service Account Creation

Also sometimes referred to as "automated onboarding," this is a key component of a scalable and efficient platform, as it does not require staff to manually create accounts. Each of the services we examined allow for this, with the exception of **M-Pesa**. Because of the need for VPN credentials and manual testing/review of results, a back-and-forth communication is required by design in their onboarding process. This results in M-Pesa having the longest lead time to integration of the services we look at.

Degree of Control

This relatively qualitative metric is meant to describe the control that a user has in customizing a product that is connected to each of these services.

Square has a relatively low degree of control, as their hardware is required for many of their solutions, so it cannot be integrated freely with other open hardware. **Android Pay** is extremely customizable within the Android framework, but is not available outside of that operating system. Overall then, this offers a medium level of control. **Stripe, Paypal,** and **Venmo** each offer a high level of customization and control. For instance, as we mentioned **Venmo** has a customizable button widget. **Stripe** and **Paypal** have a similar customizable UI feature available in them. Beyond that, each service has APIs that offer total decoupling of the features and how they can be combined, which affords much more control to the consumer of the service.

As **M-Pesa** has only recently opened their API, the use cases covered by it are strict and the onboarding process requires approval of the integration. In this sense, the developer has a low level of control in the integration. Ideally, as they open their API up further, the endpoints will become more flexible in terms of what feature sets they can support.

Hackathons, Competitions, F2F Events

A corporation can benefit from new ideas and face-to-face time with developers. To foster this type of interaction, many services offer or sponsor a conference or set of hackathons.

Stripe has been known to participate in informal payments hackathons at developer-centric workspaces. **Paypal** has also been involved in putting together hackathons, like Opportunity Hack and Battle Hack, which bring Paypal core developers together with those who are consuming their services.

M-Pesa and Safaricom list several events, competitions, and hackathons for developers on their developer page. The Safaricom App Wizz, an innovation call to partner with developers to come up with products that can plug into the Safaricom network, has been running since 2013. In 2014, Safaricom also launched a venture fund called "the Spark Venture Fund" that is offered to mobile ICT start-ups based in Kenya, either through equity investment in their businesses or in the form of other debt instruments.

It is possible that the other services have sponsored events or held invite-only conferences, but our research did not turn any others up.

When looking at the developer-centric evaluation across our candidates, we see many similarities, and a couple key differences. Stripe and Paypal offer generally more mature and flexible features, with Stripe providing a slightly better online community. This last point is a strong indicator that Stripe is directly in touch with their integrators, which means they're more likely to make decisions to improve the developer experience going forward. Stripe has checked off all our criteria so far and looks like a clear leader. SafariCom has done a lot for the developer community at large, but not as much specifically around M-Pesa. As a relatively new contender in the public API space, it has met the basic requirements of providing documentation and allowing for public integration, but could do a better job of actively engaging with the community and automating onboarding.

Business Model: Viability and Profitability

Finally, with our list evaluated thoroughly, we can take a look at the strategy and costing. This step is saved for last for a couple of reasons. First, if an API doesn't make the cut for any of the previous three evaluations, it's not worth putting any amount of money into unless there's an exceptional cause for doing so. Secondly, this costing/quoting process can often be more time-consuming and sometimes poorly documented or requires a quote, so it helps to have the list narrowed down before investing that time.

General topics to evaluate during this phase include the standard revenue agreement, the fit of the terms of service with your business model, the transactional portion of the business model, what the developer pays versus the platform, any marketplaces that help you promote your integrated solution, and the approval process for new developers. After this step, you should have a matrix that looks like **Appendix A** and be ready to make your decision.

Standard Revenue Agreement

Before making a decision, it's important for all API Consumers to consider the cost of connecting their business to an API. Charges can consist of flat fees, recurring fees, transactional fees, and administrative fees. In our analysis, each of the providers has a per-transaction cost.

Stripe has outlined a simple model for pricing in the U.S. on their website. It's 2.9% + 0.30 USD per successful transaction, with volume discounts for businesses processing more than 80,000 USD per year. **Paypal** has similar rates outlined on their website. In the U.S. it's also 2.9% + 0.30 USD per transaction for a seller, and international sales are 3.9% + a variable flat fee based on the currency. Payments Pro is an additional 30 USD/month. **Square**'s processing fees are listed on their website as 2.75% for a credit card swipe, online sale, or paid invoice. Fees for online e-commerce transactions are 2.9% + 0.30 USD. Fees for manually entered transactions are 3.5% + 0.15 USD. Additionally, there are one-time costs associated with purchasing their POS hardware. **Venmo** emphasizes on their website that all bank account and most debit card transactions are free. There is a 3% fee for sending money from credit cards and some specific debit cards, but receiving money is always free. **Android Pay** states in their FAQ that their service does not charge any fees for purchases through Android Pay, but you still pay fees to your merchant acquirer as you do with regular credit card transactions. So in essence, there is a variable transactional fee for credit cards only, depending on the type of credit card.

M-Pesa provides a tool that calculates tariffs for end users for a given transaction type and amount. For instance, it costs 15 KSH to transfer 1000 KSH to another registered user. This pricing can be useful in putting together a business model, but it is unclear from the documentation what the cost is to the API Consumer in addition to the end user. This should be as prominently listed as the other API Providers' rates.

App Approval and Distribution

Once you've made a technical integration and have evaluated the transactional costs, approval and marketing are the next considerations. Evaluating how a platform or API might help you to connect your solution to potential users could help to determine which platform to go with.

Stripe does not require manual approval for integration, so there is no friction once you are ready to launch. They do not offer a service to help promote your product, but they do have a gallery to show off some of their interesting integrations. **Paypal** also does not require manual approval and allows for launch of integrated products without notice. They also have a partner program for B2B businesses to generate client leads and drive requests from their other clients. Since **Square** primarily depends on the use of their proprietary hardware, launching of a product is not really applicable for them. Like Stripe and Paypal, they also have a partnership program to help foster growth in their small business clients. **Venmo** does not require manual approval for integrated solutions, but also does not have any official partnership program publicly listed. **Android Pay** does not require manual approval for integrated products. They feature curated lists of partners on sections of their website, and of course they have the Google Play app store for distribution.

M-Pesa has a manual approval process in order to maintain tighter control over who is using their newly opened API. Once an app is launched, Safaricom does have a marketplace set up to help to promote the services that have integrated with M-Pesa.

Now that we've covered most of the business case, have a sense of how much a solution will cost to run, and how it looks from a technical perspective, we're ready to take a look at all these factors to make a decision for the partner we wish to integrate with.

Step 4: The Decision

This last step is typically subjective, as it comes down to the most important aspects for each API Consumer's business model, and their product roadmap going forward. There may be choices between the cheaper integration in the short-term and the experience that's more efficient for the developers in the long run. One API may have a great new feature, but could also be more difficult to implement than another. Whatever it may be, there is not necessarily one right answer here.

That said, after reviewing each of these factors, Stripe and Paypal are quite clearly the best fit for each of our archetypical API Consumers looking to integrate a payments feature. These two services provide the most features at competitive pricing, document them thoroughly and publicly, allow for automated onboarding and launch, provide the best developer experience, and even assist in the process of promoting the product. Additionally, and perhaps most critically, we've found that each of their integration processes take 1-2 hours, versus up to 2 to 6 months for less open financial API services like M-Pesa.

Step 5: Monitor the Ever-Evolving Market

At this point, we've made our decision, but that does not mean that we're done with our evaluation forever. Because of our currently unmet needs and the rapidly developing nature of the payments API space, we must continue to keep an eye out for future products and revisit and revise our matrix as the market's offerings change.

In more developed markets, for example, there have been a number of efforts to open up the payments space via APIs, each building off the previous generation. Starting with PayPal in the 1990s and continuing through the present day's drop-in digital solutions like Stripe and plug-in physical solutions like Square, there is a continued push for more and more openness.

In emerging payments markets such as Kenya, we can see the beginnings of such an evolutionary process in the works. Many times, however, this is in the form of clever hacks that stretch the limits of what an API was primarily intended to do, which results in a less elegant, reliable, and robust service.

As one example, during our research and compilation of this report, Stripe launched their aforementioned Atlas product, which drastically reduces constraints for integration with their API from previously excluded countries including Kenya.

Unfortunately for mobile money services in Kenya, Paypal and Stripe still each require an intermediary UI or an agent to manually move money between an m-wallet, their e-wallet, and physical cash. By opening up their API fully, M-Pesa has an opportunity to use their advantages in this regard to potentially leapfrog the competition that is rapidly growing deeper into the Kenyan market.

For our analysis in particular, our API Consumers will want to keep an eye on M-Pesa to see if they decide to evolve their Public API into an Open API with all of the associated benefits, and whether they will begin to offer ReSTful JSON services. They'll also want to spend some effort monitoring new startups to see if any Open API solutions arise in the meantime.

Rationale for Open APIs in Financial Inclusion

By now we've talked a lot about open APIs here, and even given some of the latest examples so far. But what are the major benefits for the industry at large? Why would I want to connect with an API? Why might I want to provide my own service as an open API? These are great questions that we've hopefully provided some answers to already. In the interest of providing these more explicitly, we enumerate some of the concrete reasons below.

Key Takeaways

1. Financial providers can benefit from the efficiency and added growth potential of open APIs and collaboration. Even legacy banking systems can be made state of the art by connecting through middleware to provide modern, open solutions.
2. Developers benefit by leveraging existing services in order to focus on making their core services the best they can be. In particular, startups can increase their chances of success by reducing overhead in producing a minimum viable product.
3. By reducing costs in this manner, opportunities for social and financial impact where the margin was previously too low to consider become viable business opportunities.

Benefits for Financial Providers

Financial providers include services such as banks, private lenders, credit card networks, mobile money payment systems, and aggregators. Simply put, they provide the backbone that allows money to get from point A to point B when needed.

On a high level, several policy papers have been released that highlight the values of interconnectedness and openness in the banking community. A report put out earlier this year by the EBA Working Group on Electronic and Alternative payments stated, "An acceleration of the digital economy is foreseen when the financial industry succeeds in creating the next level of digital interfacing."

Another report put out in late 2014 by the Open Data Institute and Fingleton Associates how competition and consumer outcomes in UK banking could be affected by banks giving customers the ability to share their transaction data with third parties using external APIs.

It illustrates how outcomes would be improved if banks published this non-personal data as open data, including reduced human effort, security, expanded opportunities, and more.

Closed core banking and credit network services have historically been the preferred means to make this happen, as they provide a sense of security by limiting access. However, as we've briefly discussed earlier, this comes at the expense of convenience and speed, and a peer review process of the security mechanisms. Opening up these systems and allowing more players into the game can substantially reduce friction in the process and lead to systems that are truly secure as opposed to providing a sense of security primarily through obfuscation.

In the United States, services like Check 21 have taken the more antiquated Automated Clearing House (ACH) standards (based on magnetic tape storage) and given them a much-needed facelift. By leveraging the available technologies and working to adjust the federal policy, this transition fully digitized the personal check and took processing times from two or three days down to thirty seconds.

In Kenya, the main banks have started using M-Pesa's APIs for their interbank transfer needs for similar reasons. M-Pesa is seen as favorable over the banks' own core banking systems purely because of the speed and convenience that is offered. M-Pesa keeps a certain balance of float at each major bank in order to facilitate their own internal business cases. At some point, they realized that by keeping a real-time digital ledger and taking on the burden of a delay themselves rather than passing it to the consumer, they were able to create a model that worked best for everyone involved. Though they didn't initially see that benefit feeding back to the banks themselves, the open aspect of their model ultimately facilitated this.

Benefits for Developers and Social Enterprises

The benefits that exist for developers should be clear already, but are worth stating explicitly. APIs allow a developer to focus on creating custom logic around their own business cases, and to "abstract away" anything that falls outside of that. By using payments APIs and social sign-in, for example, an site built to foster a Micro-Finance Institution can focus on collecting data to inform terms and rates of their loans their customers are looking for, rather than worrying about building authentication and payments engines from scratch.

For the developers actually building and exposing these sorts of APIs, the benefits are mostly in the form of reusability. By building each service as a general API endpoint, instead of building out the same functionality repeatedly for each client that is brought on board, they can provide the documentation to each new client. Furthermore, these endpoints can be combined in unique ways for different use cases that may not have been foreseen at the time of making them public.

Benefits for the Under-Banked

There is a large and growing number of startups and mature financial institutions that are interested in financial inclusion as a legitimate part of their business model. Currently in

many of these markets, each company is building out their own custom code for the same business cases. This is an inefficiency that could be mitigated in large part by open APIs.

By providing platform tools to access services that are needed by virtually everyone in the space (e.g. credit scores, payments across bank accounts and mobile wallets, KYC services, bulk SMS, etc), each company is able to focus on their particular strengths. This means higher-quality, less-diluted products for everyone, ultimately providing a stronger array of options for the population at large.

Broader socio-economic benefits

Beyond the financial sector, there are higher-level benefits of this sort of modularization of an industry. The fundamental building blocks that result from open APIs working with each other leads to more rapid growth and understanding of the problems at hand. The collaboration that results in second and third level solutions built from these fundamental services allow the innovators and entrepreneurs to shed the burden of re-inventing the wheel for each solution, and instead reach for the true frontier of what technology is capable of.

In the developing world, as much as anywhere, there exist a great many inefficiencies and problems in need of a solution, and many of these are not profitable for a business that counts proprietary data and services as a competitive advantage. Contrary to these classically popular viewpoints, opening up services for collaboration and increased efficiency reduces costs and exposes newly sustainable business opportunities.

Recommendations for Next Steps

As a result of the research, we've synthesized several points as suggested next steps for the industry at large. These points are actionable for financial service providers, financial technology and inclusion funders, and generally anyone interested in entering the space.

Bring Cutting-Edge Features to the Developing World

As we saw in the analysis above, there are some notable advantages of Safaricom's M-Pesa solution and the like. However, we also noted that this service lacks some of the more open features that mature alternatives like Stripe and Paypal offer. The fact that the more mature services currently don't provide all their solutions in all geographic regions that could utilize them means there is a gap to fill in the meantime.

Taking a look at M-Pesa-like services and encouraging them to add more modern features is an important step that must be taken. We can encourage the growth of integrated services by automating the onboarding and integration processes, developing online documentation with code samples, and replacing technical barriers like manual VPN setup and tightly coupled SOAP services with something more efficient. In short, it's important to prioritize making the "M-Pesas" of the world as scalable and accessible as the "Stripes" of the world.

Key Players Take the Lead

Wherever possible, a critically important step to be taken is for key players in the market to act as flagship examples of what's possible when interfaces are opened up. By demonstrating that they are willing to evolve, they set a precedent, proving it is an achievable and practical goal. By providing this precedent, they inspire others to approach their unique challenges with the same strategy of openness and collaboration.

Industry leaders should regularly follow and revisit the evaluation steps above to see if they're ready for the next evolutionary step as a service provider. Investors and other funders in the space can help to encourage this behavior as well by setting openness metrics as part of their funding.

In Kenya, while M-Pesa's newly public API will not be the final solution for the payments space, it is a step in the right direction. In order to stay current, competitors will have to follow suit but must also continue to innovate independently in order to maintain competition, which will lead to sustained and organic growth from within the industry.

Promoting Middleware Solutions

As discussed previously, many big players will complete an analysis and decide that openness is not the best decision for them at this time, or that preparing their services as a fully open API may take longer than demand dictates.

For instance in Kenya, both the banks and MNOs have made strides in public APIs, exposing their services a certain amount for limited services. However, there exists a gulf in between the two types of entities for those who want to integrate banking with mobile money. Middleware services can bridge that gulf by building the translation layers between the two, providing a platform for more services to build off of.

Building or investing in these types of services will encourage a new phase of growth by providing these building blocks and filling gaps in the supply of these services.

Fostering a Developer Community

Of course, the main ingredient in building an open, interconnected ecosystem of platforms and APIs is of course the developers needed to build and connect them. Creating and maintaining collaborative incubator and accelerator spaces, online forums, offline conferences, and hackathons can help provide a catalyst for developers to get together and share knowledge and experiences. Creating a true community that fosters collaboration around a given space or skillset can have a substantial impact.

Large industry players and their funders should come together to provide resources to up-and-coming developers in the form of incubators, accelerators, and other creative spaces. This can only serve industry in a positive way.

Sharing Knowledge and Experiences

Engineering blogs, research reports, and other forms of publicly accessible documentation are a wonderful tool for providing guidance based on experience, and helping others avoid pitfalls. With this information published on the web, new startups in the market can leapfrog some of the learning curves and other hurdles often associated with getting off the ground.

By providing funding for broad, industry-wide studies as well as deep dives into some of the major industry players and up-and-coming innovators, we can learn about ourselves as an industry and more effectively learn how to efficiently grow the entire space.

In addition to general knowledge transfer, it is advantageous for larger companies to open source key parts of their product that are not critical to their bottom line. By doing so, they can watch smaller players develop innovative solutions that would otherwise not have been realized. In this sense, the larger groups can effectively outsource some of the cost and risks that would otherwise come with research and development.

In the U.S., companies like Stripe and Square are great examples of these concepts. Stripe has a great engineering blog that documents their latest features, partnerships, technologies, and more. Square also has a detailed and well-used engineering blog, but additionally has a huge amount of open-sourced software libraries for developers to use. Ranging from Android libraries for making API requests in general, to server-side tools for rapid deployment of services, they share their expertise in order to foster growing companies' growth and ultimately their chance for a partnership.

In Kenya, on the open-source side of things, there is a groundbreaking library called PesaPi, which is a single piece of open source software that connects developers to many of the mobile money services across Africa. We also see that Safaricom offers a blog that often includes content about M-Pesa, but is typically limited to marketing and business-level content and rarely includes technical level. These two examples are a great start, and a sign that things are moving in the right direction, but encouraging more of this behavior would only be beneficial to the industry.

Conclusion

Our discussion started with an exploration of interfaces - both UIs and APIs - in their various forms. We continued by describing how a growing digital financial service provider may choose to evolve their own interfaces from a basic product by producing a modular set of replicable features to be shared with trusted partners, and eventually opening an API to the broader industry. The organization proceeds through this evolution by determining that risks like reliability, security, and brand recognition are outweighed by rewards like increased user base, scalability, and partnering opportunities at each step. It is critical for growth that an organization knows how to mitigate these risks through a proper analysis, and has the capability to fully understand the rewards.

We explored a few common examples of businesses operating in Kenya, and how they might benefit from their service providers further evolving in the manner described by opening up their APIs. The M-Commerce site needs automated, timely onboarding so they can quickly launch and start drawing in revenue. The Mobile Money Distributor requires a payment history API so they can reallocate resources currently dedicated to scraping SMS messages to something more productive. Mobile App Providers need more interactive documentation, along with code samples, SDKs, and embeddable libraries, so they can quickly understand and integrate digital services as smartphones become more prevalent. The Subscription Service requires a recurring subscription feature that is not publicly provided as an API endpoint. The Money Management App has no way to aggregate payment histories across services, as there is no user-centric authentication in place to share one's own financial data.

To understand on a more detailed level what is needed, we defined a method to compare a set of APIs from perspectives of the business as well as the developers within it. By comparing M-Pesa with its counterparts in other regions, we came up with a list of constraints and benefits that each provider had. We noted some API trends that came out of this analysis for the developed and developing world.

While significant progress was made in evolving the M-Pesa set of products from UIs and Private APIs to a Public API, there is still significant work to be done to satisfy the needs listed above. Evolving to an Open API with the mindset of becoming a platform would go a long way in satiating businesses' needs.

These critically missing pieces can be taken as either far-off roadmap items for existing providers, or imminent opportunities for up-and-coming startups to gain a competitive edge. Until one or the other solves these shortfalls, developers will be held back from realizing the full potential for their products, and customers will be left with imperfect services.

To this end, we presented recommendations for those with the power to encourage more openness. The industry - financial providers, developers, social enterprises, and even under-banked populations – stands to benefit from such a reshaping of the landscape. Bringing high-tech features and talent to the developing world is critical. Setting up key players in a market to adopt these features, and promoting middleware solutions to supplement in the meantime, is a necessary component of that strategy as well. Lastly, but perhaps most importantly, providers must continue to foster a local developer community to make it all happen, while sharing all knowledge and experience along the way so successes can be replicated while mistakes are rarely repeated.

The recipe for a successful and healthy digital economy consists of, and even requires, each of these key components to be readily available to everyone in the industry. We are excited to see who will pick up the challenge, accelerating the process of creating and maintaining these critical elements to create a thriving ecosystem around them.

Acknowledgements

Of the many organizations that we spoke with in researching this report, we'd like to explicitly acknowledge several of the key resources that assisted us in our efforts. Their willingness to make themselves available and to provide candid responses contributed substantially to the perspective presented here:

- Safaricom and M-Pesa: http://www.safaricom.co.ke/personal/m-pesa
- MobiDev: http://devs.mobi/
- Umati Capital: http://www.umaticapital.com/
- Caytree Partners: https://www.caytree.com/
- M Changa: http://changa.co.ke/
- Kabla Incubator: http://kabla.co/
- People Plus and Uhabisu: https://www.uhasibu.co.ke/

Contributors

Contributors to the content of the report, and their respective roles follow:

- Matt Grasser, Report Editor
- Asa Nyaga, Researcher
- David del Ser, Project Manager
- Ignacio Mas, Advisor

Appendix A: Full Comparison Matrix

Features and Benefits

	M-Pesa	Stripe	PayPal	Square	Venmo	Android Pay
B2B	M-Pesa "G2" API	Stripe Connect API	Paypal Adaptive Payments	Not Available	Not Available	Not Available
B2C (Payouts)	M-Pesa Bulk Payments	Stripe Transfers API	Paypal Mass Payments	Not Available	Deprecated	Not Available
C2B (Accept Payments)	IPN / Bank to Wallet	Stripe Checkout	Paypal Checkout	Square Reader, Square Stand, Square Register	Venmo Transfer	Android Pay
Bill Pay and Invoices	Lipa na M-Pesa	Stripe API	Paypal Request Payment	Square Invoice	Not Available	Not Available
P2P	M-Pesa Transfers	Stripe Connect API	Adaptive Payments	Square Cash	Venmo Transfer	Google Wallet
Marketplace	M-Pesa Merchant API	Stripe Connect API	Braintree Marketplace	Not Available	Not Available	Not Available
Financing	mShwari	Stripe + Kabbage	Paypal Working Capital	Square Captial	Venmo + Ledge	Not Available

Infrastructure and Tech

	M-Pesa	Stripe	PayPal	Square	Venmo	Android Pay
SOAP	Available	Not Available	Available	Not Available	Not Available	Not Available
Rest	Not Available	Available	Limited Availability	Limited Availability	Available	Not Available
Webhooks	Not Available	Available	Available	Available	Available	Not Available
Security and Auth	1. API Credentials 2. TLS 3. VPN	1. Oauth 2 2. TLS	1. SOAP/NVP Credentials 2. REST Outh 3. TLS	1. Oauth 2 2. TLS	1. Oauth 2 2. TLS	1. API Credentials 2. TLS 3. Oauth
Fraud Protection	Not Available	Available via Fraud Filters	Available	Basic Guidelines	Basic Guidelines	Available
Data & Reporting	Not Available	Available	Available	Available	Limited	Not Available
High Availability	No SLA Available	No SLA Available	No SLA Available	No SLA Available	No SLA Available	No SLA Available

Developer Experience and Outreach

	M-Pesa	Stripe	PayPal	Square	Venmo	Android Pay
Documentation	Available	Available	Available	Available	Available	Available
Online Forum	Not Available	Available	Available	Not Available	Not Available	Android Developer Group
Libraries and SDKs	Not Available	Available	Limited Availability	Not Available	Available	Android SDK
Support	Email Support Telephone Support	Online Support Only	Online Support	Online Support Only	Online Support Only	Developer Group
Premium Support	Not Available	Not Available	Not Available	Not Available	Not Available	Not Available
Self-Service account creation	Not Available	Available	Available	Available	Available	Available
Degree of Control	Low	High	High	Low	High	Medium
3rd Party Ecosystem/Integrations	Avilable	Available	Available	Available	Available	Available
Hackathons, Competitions, f2f events	Not Available	Informal Hackathons	Sponsored Hackathons	Not Available	Not Available	Not Available

Business Model

	M-Pesa	Stripe	PayPal	Square	Venmo	Android Pay
Standard Revenue Agreement	Per Transaction	Per Transaction	Per Transaction	Per Transaction	Per Transaction	No Added Fees
Developer Pays	All	All	All	All	All	Credit Card Fees
Freemium	N/A	N/A	N/A	N/A	N/A	N/A
Platform Pays	N/A	N/A	N/A	N/A	N/A	N/A
Platform Subsidies	N/A	N/A	N/A	N/A	N/A	N/A
App Approval Process	Manual Approval	Open	Open	Not Available	Open	Open
App Marketplace	Safaricom App Store	Stripe Gallery	PayPal Partners	Square Partner Program	Not Available	Featured Partners and Google Play

Appendix B: Further Reading

"APIs: The Building Blocks of the App Economy." NY Times. CA Technologies, n.d. Web. <http://paidpost.nytimes.com/ca-technologies/apis-the-building-blocks-of-the-app-economy.html>.

Data Sharing and Open Data for Banks: A report for HM Treasury and Cabinet Office. N.p.: Open Data Institute and Fingleton Associates, September 2014. PDF. <https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/382273/141202_API_Report_FINAL.PDF>

Feng, Amber. "Building Stripe's API." Amber on Rails. N.p., 23 Feb. 2013. Web. 14 Dec. 2015. <http://amberonrails.com/building-stripes-api/>.

Jotwani, Amit. "Unpacking Developer Experience." Slideshare. N.p., n.d. Web. 14 Dec. 2015. <http://www.slideshare.net/ajotwani/unpacking-developer-experience>.

Lee Cohick, Jeremiah. "Elements of API Excellence." Speaker Deck. N.p., n.d. Web. 14 Dec. 2015. <https://speakerdeck.com/jeremiahlee/elements-of-api-excellence>.

Moszczynski, Michael, and Jacob Korenblum. "Why Open APIs Matter: Tech Partnerships Power Development." CGAP. N.p., 8 June 2015. Web. 14 Dec. 2015. <http://www.cgap.org/blog/why-open-apis-matter-tech-partnerships-power-development>.

Opinion Paper on Exploring the Digital Customer Services Interface. N.p.: EBA Working Group on Electronic and Alternative Payments, 11 May 2015. PDF. <https://www.abe-eba.eu/downloads/knowledge-and-research/20150511_EBA_Opinion_Paper_on_exploring_the_DCS_v1_0.pdf>

Spencer, Travis, Andreas Krohn, Rhys Fisher, and Mark Boyd. "API Platform Defined: When an API Provider Is a Platform." Nordic APIs. N.p., 05 Aug. 2014. Web. 14 Dec. 2015. <http://nordicapis.com/api-platform-defined-api-provider-is-a-platform/>.

Tellez-Merchan, Camilo. "Can Open APIs Accelerate the Digital Finance Ecosystem?" CGAP. N.p., 01 June 2015. Web. 14 Dec. 2015. <http://www.cgap.org/blog/can-open-apis-accelerate-digital-finance-ecosystem>.

Thomas, Andria, and Marcus Watson. "Partnership: Missing Ingredient to Mobile Money APIs." CGAP. N.p., 19 Aug. 2015. Web. 14 Dec. 2015. <http://www.cgap.org/blog/partnership-missing-ingredient-mobile-money-apis>.

Yu, James. "Designing Great API Docs." Parse Blog. N.p., 10 Jan. 2012. Web. 14 Dec. 2015. <http://blog.parse.com/learn/engineering/designing-great-api-docs/>.

Appendix C: Key Points Summary

1. Interfaces, while typically associated with an interactive visual representation like a mobile device or a website, comprise more than an end user interface.
2. The wide array of interface types that exist are tailored to a specific set of parties involved at either end of the interface. User Interfaces translate digital information to humans, while APIs exchange data between machines.
3. The distinction between API types is typically around the level of access that is provided to the data, and how that access is administered.
4. Businesses that enter the digital space tend to evolve in a relatively predictable way, with certain criteria to evaluate at each transition.
5. It is important that companies treat these evaluation points carefully and consider the implications of using a digital service in the context of their product roadmap and growth strategy.
6. In the digital domain, developers are the people building inter-business relationships.
7. To ensure these relationships are sustainable, it is critical that developers are recognized as a key component of the growth and success of a company looking to thrive in the digital space.
8. Concepts like Developer Experience (DX) and other technical criteria provide a great set of rules to evaluate prior to integration.
9. Having a strategy for evaluating potential technology partners against each other is critical. We suggest several high-level steps: Explore, Evaluate, Decide, Monitor
10. As an API Consumer, having a strategy for evaluating API Providers against your needs is critical. We suggest several high-level components for this evaluation: features and benefits, infrastructure and technology, developer experience, and business model.
11. Each step should help narrow down the available options, but ultimately the decision is likely to come down to an evaluation of the top candidates against business needs.
12. Continuing this evaluation as the market evolves is a critical step. Solutions that you previously “hacked together” because of technological constraints may be made more elegant or robust by a new feature or service that’s been introduced since the last evaluation.
13. While M-Pesa has significantly progressed in their evolution from a semi-private IPN to a public API, they still lack some of the benefits of the more mature open APIs.Financial providers can benefit from the efficiency and added growth potential of open APIs and collaboration. Even legacy banking systems can be made state of the art by connecting through middleware to provide modern, open solutions.
14. Developers benefit by leveraging existing services in order to focus on making their core services the best they can be. In particular, startups can increase their chances of success by reducing overhead in producing a minimum viable product.
15. By reducing costs in this manner, opportunities for social and financial impact where the margin was previously too low to consider become viable business opportunities.

Appendix D: Anatomy of an Interface

What do interfaces look like to a developer?

An interface provides the ability for a certain audience…

API / SDK	UI
Target users of a *service*: • Authentication providers • Automated processes • Financial Services	Targets users of a *product*: • Smartphone • Merchant POS Systems • Web surfers

…to communicate over a given protocol…

ReST	SOAP / WSDL
GET /payments/{id}	<definitions name="GetPaymentService" targetNamespace="http://a.test/HelloService.wsdl" xmlns="http://a.test/wsdl/" xmlns:soap="http://a.test/wsdl/soap/" xmlns:tns="http://a.test/HelloService.wsdl" xmlns:xsd="http://a.test/2001/XMLSchema"> <message name="PaymentsRequest"> <part name="paymentId" type="xsd:string"/> </message> <message name="PaymentsResponse"> <part name="id" type="xsd:string"/> …

…using a given language.

JSON	XML

{
"id": 1,
"label": "MNO Topup",
"price": 5000,
"tags": ["data","acct1"]
}
	<?xml version="1.0" encoding="UTF-8"?>
	<root>
	<id>1</id>
	<label>MNO Topup</label>
	<price>5000</price>
	<tags>
	<element>data</element>
	<element>acct1</element>
	</tags>
	</root>

The example above shows what a request to an API to list a payment by id might look like. We've illustrated a request to list a payment with an id of 1, by communicating over a given protocol, and have received a response. The response includes some attributes of that payment including the id, a label, the price of the transaction, and some tags that are associated with the payment. This response could then be represented in a product's UI as a "payment details" screen.

A high-level overview of how a developer might look at and interpret certain technical features of an API. This is by no means an exhaustive set of criteria, but is provided with the intent of giving some sense of what these features look like.