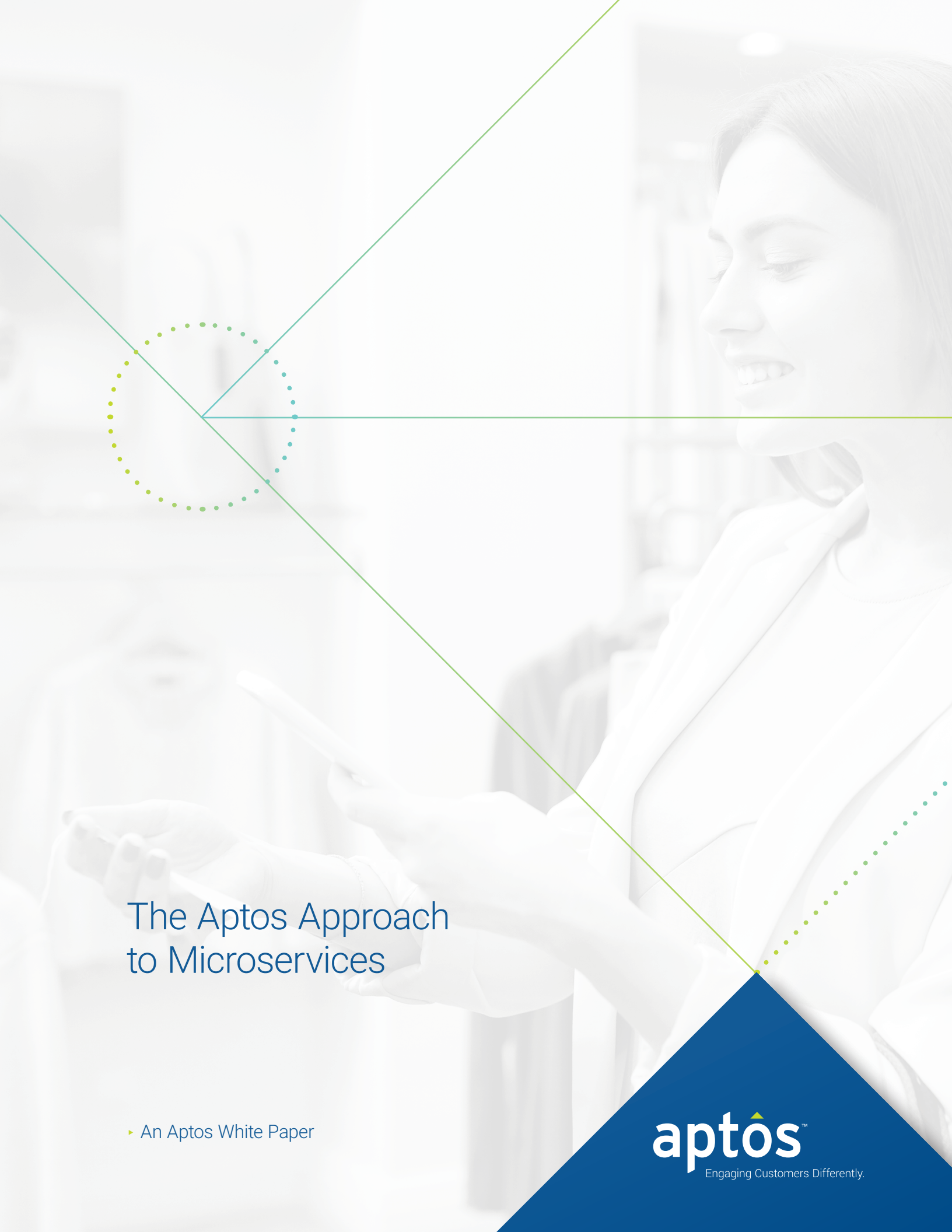


The Aptos Approach to Microservices

An Aptos White Paper

aptos™
Engaging Customers Differently.

Table of Contents

Executive Summary

"Microservices" has become the rallying cry of technologists everywhere, and a buzzword du jour, but the term is often used far outside the original intended definition. This document explains the general architectural principles of microservices as Aptos embraces them, and how they are specifically applied in support of the Aptos ONE platform.

While this document is technical in nature, it is important for business leaders to understand that a microservices architecture has the potential to drastically change how they purchase, consume, and maintain the technology solutions that enable their business. The business goal is cost-effective speed and flexibility, so that retail companies can keep up with rapidly evolving consumer expectations.

Microservices have somehow come to mean the penultimate future of enterprise software, and the retail industry is not immune to the enthusiasm. However, microservices are not a specific development language or methodology, nor are they a catch-phrase for solutions that are generally on a "modern technology stack." Microservices form the basis of a specific architecture design. As such, the "rules" of microservices are more like guidelines. This makes the reality of microservices much more nuanced and open to debate – an art, rather than a science. It is not a panacea, and there may be domains where it is simply not a realistic technical architecture.

Aptos has applied six key principles to our approach to a microservices architecture, but most importantly, we have taken the opportunity afforded by our approach to envision and truly enable an omnichannel-native future for retail. This vision is embodied in Aptos ONE, our platform for retail microservices.

Understanding the Aptos approach to microservices in general and Aptos ONE specifically is key to bringing microservices into your enterprise. But our platform alone is not enough to ensure that your own microservices strategy will be a success. We highlight important implications of a microservices approach, with recommendations on focus areas as your own organization tackles this new architecture.

Why Is Everyone Talking About Microservices?

Microservices have the potential to drastically transform how companies purchase, consume, and maintain technology solutions that enable businesses. It is a solution that - whether they realize it or not - retailers have long been waiting for. As consumer behavior rapidly shifts, as new channels open up, as new data about consumers and demand become available, retailers are finding that their organizations need to now support a pace of change that is well beyond their ability to keep up. And the main culprit for this challenge is their technology infrastructure.

The technology footprint of most retailers today has grown out of a mix of packaged applications that are so customized and outdated they are no longer supported by the original vendor; custom code and hard-coded integrations; and a sprinkling of modern technology solutions to support digital channels. The architecture, taken together, has become so brittle that the end result for the business is long lead times for new capabilities, high costs for new capabilities that are integrated into enterprise systems, and a high risk of failure for the projects that do get approved.

Done well, a microservices architecture address all of these pain points. It enables:

- **Speed to market.** A microservices architecture enables retailers to quickly deploy new capabilities in a highly cost-effective way, and to continuously improve capabilities independently of major upgrades. This results in much faster access to evolving solutions.
- **Flexibility.** Flexibility here can be measured in terms of the retailer's ability to work with partners to innovate within the bounds of the architecture, flexibility in choosing how or where to deploy new capabilities, and flexibility in quickly changing or adapting how microservices are orchestrated to enable new processes.
- **Lower total cost of ownership.** The flexibility of microservices enables retailers to take advantage of four major areas of cost savings in technology deployments. Fewer developer hours should be required to make code changes, because of efficiencies gained in maintaining and evolving small, isolated, and well-understood pieces of code. Testing costs are significantly reduced, because one single change in functionality does not, by design, require testing all of the integration points for the entire solution. Integration is faster and standards-based, eliminating the cost and time of maintaining hard-coded integration points. And infrastructure costs are also significantly reduced, because each microservice is independently scalable. One service's demand for resources does not require scaling the hardware to support the entire solution. Businesses can scale technology to meet specific spikes in demand, and can scale down – saving infrastructure costs – when that scale is not needed.

- **Greater IT/business alignment.** Retailers typically upgrade their software solutions when they need access to new innovation, or when their business processes have changed significantly and they need their technology to adapt to support the changes. When retailers deploy monolithic software applications, with all of the challenges identified above, they find they have to wait until the process misalignment with technology is so painful, or the innovations they are missing out on become so imperative, that they can finally make a business case for the investment required to implement the upgrade. The flexibility and continuous improvement of microservices takes all of that off the table. Retailers leveraging microservices should be able to adapt their technology to business process change as it happens, eliminating the business case challenges – and time delays – that have kept them from alignment in the past.

- **Greater ability to innovate.** A microservices architecture is not the only path to retail innovation, but they it is a faster and more stable path. Retailers often undertake experiments in customer experience or services, but because of their brittle technology infrastructure, they often position these experiments far outside the enterprise, and with little path back into operations. The end result is a lot of great ideas that are often successful, but not so successful that they can overcome the hurdle of ROI required to make the integration changes needed to make innovation a regular, supported part of the enterprise. Microservices are designed to enable a platform for innovation – they tackle the integration problem head on, and if a small experiment is successful, it is simply a matter of moving the experiment into the live environment to deploy it at scale.

In order to take advantage of these benefits, retail companies – including business leaders – must understand what a microservices architecture really means, and how to apply it. Companies like Amazon and Netflix are already well along the microservices learning curve, and it forms the foundation of these companies' ability to innovate. To remain competitive, retailers must learn to use microservices to their own advantage.

What Are Microservices?

A Google search for microservices will yield a dizzying array of results. There are so many things that have been attributed to microservices that it may be helpful to examine what they are not:

- **Microservices are not a programming language.** In fact, microservices try to enforce flexibility by encouraging the use of any language – the objective is to select the right language to fit the need. The same is true of database selection. What this means, however, is there are no hard and fast rules or toolsets that you can select that automatically mean you are developing microservices.
- **Microservices are not a development methodology.** While a methodology may be less specific than the rules and syntax associated with a programming language, it still implies more structure than the principles that define microservices. There are no specific steps to follow that automatically lead to microservices, whether starting from scratch, or converting an existing monolithic solution into a set of microservices.
- **Microservices are not just a general term for any solution deployed on a "modern technology stack."** Microservices are often spoken about in the context of cloud deployments. But hosting an application in the cloud does not automatically make an application microservices-based. Microservices and cloud go very much hand-in-hand, along with the container solutions that make managing microservices possible. Together with modern development operations, microservices, containers, and cloud infrastructure combine to create "cloud-native" solutions. But not everything in the cloud is truly cloud-native.

An Architectural Style

The term "microservices" is most often used in conjunction with "architecture". Microservices are bundles of code that operate independently of each other, but more importantly can be deployed with far more variability in the infrastructure than a monolithic application. In order to achieve that kind of flexibility, developers must follow a set of principles. These principles are not hard and fast. Even when the principles are well understood, they are often difficult to put into practice, and developers are encouraged not to expect to get it right the first time. There is as much art as there is science in developing a microservice.

The General Principles of Microservices

Leverages service-oriented architecture. Service-oriented architecture (SOA) has been around for years, and is a well-understood software design. But original SOA principles were developed pre-Internet, and pre-cloud, and included a heavy reliance on enterprise service buses that rapidly became more of a hindrance than a help. Microservices builds on the principles of SOA, but updates them for a network- and cloud-enabled world. And while SOA principles are well understood, microservices create new implications for software development and management that go far beyond the complexities originally anticipated by SOA architecture.

Uses only APIs, over standard network connections. APIs, or application program interfaces, make integration between applications easier and more flexible than hard-coded integrations of the past. Organizations make it easy for others to leverage their applications by publishing their APIs so that anyone (with the right security) can access the capabilities their solutions enable. For example, when you create a route to a destination in Google Maps, and it offers to summon an Uber to get you there, Google and Uber are sharing information and leveraging each other's capabilities via APIs over standard network connections (the Internet). Google didn't need to spend time with Uber's developers in order to offer the integration, any more than Uber needed to spend time with Google's. And if Uber offers a new kind of service (say, air taxis), Google doesn't have to reimplement the integration in order to be able to expose that new service to end users. That is the value APIs can provide.

Technology stack independence. Furthering the Google Maps/Uber example, it also does not matter whether Google and Uber are using the same development languages, databases, application servers, or even respective cloud infrastructure in order for their applications to talk to each other. Historically, developers have made tradeoffs in choosing the technology stacks for their solutions, especially when creating complex solutions that, for example, need to combine optimization, transactions, and what-if analysis. Microservices architecture actively encourages technology independence, guiding developers to select the best programming language and database for each microservice, depending on the service's purpose, without concern for impact on inter-operability.

Elastic. Microservices architecture is designed to create opportunities for each microservice to scale individually. This has tremendous impact on cost of ownership. In a monolithic environment, the infrastructure would have to be sized for the entire application's peak utilization. For most retailers, that's usually one day in late November or December. But that infrastructure has to be bought and paid for, even if it sits unused the whole rest of the year.

In a microservices architecture, individual microservices scale up as needed. For example, if a retailer's online site was experiencing heavy volumes, but they were all known customers who were logging in to complete a transaction, the services that govern user authentication and loading default shipping information into an order might have to scale up, while services that govern address verification – which might be needed more heavily for new customer sign-ups – do not. Retailers only pay for what they use, and only when they use it.

Resilient. In a microservices environment, resilient means that if one microservice fails, it should not bring down all the rest of the microservices that are working together to deliver an application. This is one of the biggest differences between microservices and monolithic architectures. One of the reasons why monolithic applications take so long to change is because making major changes to one part means that the entirety of the application has to be tested thoroughly to make sure there are no unforeseen consequences that bring down the whole thing. With microservices, each service is wholly independent. If you need to replace it, you replace it – with little to no impact on the application as a whole.

Minimal AND Complete. Finally, and this is perhaps the most important aspect to defining a microservice, each microservice must be both minimal and complete. The tension between defining the smallest possible set of functions, while making sure that the set is not so small that it cannot effectively operate without relying on other microservices, is probably the most difficult part of mastering a microservices architecture.

One implication of the minimal and complete principle is the need to minimize duplication of code. For example, if one action that two different microservices needs to take is to send an email notification, then instead of copying the email notification code into two different microservices, it makes more sense to break out email notification as its own microservice. But arriving at that level of understand of the business and its business processes can often require multiple iterations.

Not A Panacea

Microservices is but one style of technology architecture. There may be domains where it is simply unrealistic to break up applications into smaller services. In that case, a retailer may run a hybrid environment where some monolithic applications function as very large services within a microservices environment.

And operating a microservices architecture is not without its own challenges:

- **Minimal AND complete is hard.** It can take multiple iterations to get right, and it requires a level of understanding of business process that forces developers to engage at a much deeper level with the business organization.
- Retailers who went through a recent period of attempting to stamp out all unnecessary databases will now find themselves not only having to embrace the idea of a database for every microservice, but also the idea of "eventual consistency". Because microservices operate independently from each other, the data governing an application's output may require some time to get back in sync in order to have a single, up-to-date view of the output. While there are provisions for specific kinds of transactions, within the bounds of a single service, for an enterprise view of data, eventual consistency will have to become the norm. For "single truth" purists, this can be difficult to accept.
- **Service management becomes more complex.** Functionality deployed within a monolithic architecture does not have to "discover" other functionality around it. But in a microservices architecture, extra attention needs to be paid to the management (including metadata and versioning), discovery, and orchestration of microservices.
- **Coordination between microservices becomes a critical skill.** In monolithic applications, the connections between different functions are hard-wired. In a microservices environment, connections are loosely-coupled by design. This requires a new way of thinking about messaging between microservices that is different than the expectations within a monolithic environment. This also requires new design patterns that engineers must learn in order to work effectively in these architectures.
- **Application performance monitoring and management also becomes critical.** There are two areas of focus here. In developing microservices in a modern DevOps organization, the focus on creating continuously-deployable solutions puts a lot of pressure on testing and quality assurance functions, with an emphasis on creating more automated ways of ensuring deployable software. In deploying microservices, diagnosis of errors or bugs requires a different mindset than in traditional support mechanisms, with an emphasis on more granular event monitoring, alerting, and aggregated, potentially asynchronous logging mechanisms.
- **Deployment methodology needs to become faster and more flexible**, and this will require significant changes in the speed of a retailer's IT department. In a microservices architecture, the days of building up a long list of changes and implementing them once every two years will have to go to the wayside. Half of the benefits of microservices can only be accessed if retailers keep their applications up to date at least monthly, if not weekly. This is probably one of the most radical implications of moving to microservices. But part of the reason why it's necessary is because the business can no longer evolve in step-level changes in capability delivered years apart. They need the latest and greatest capabilities as soon as they become available.

The Aptos Approach to Microservices

In addition to the general principles of microservices, Aptos has identified six key pillars that are central to our approach to a microservices architecture, but most importantly, we have taken the opportunity afforded by our approach to envision and truly enable an omnichannel-native future. This vision is embodied in Aptos ONE, our platform for retail microservices.

The Key Pillars of Aptos ONE

Cloud-Native Microservices. Just taking an existing monolithic application, adding in some APIs and throwing it into the cloud does not give you access to all of the benefits of cloud architecture. Cloud has been around for a long time, but it has only been since the arrival of microservices that technologists are truly able to take advantage of the benefits that cloud can provide. Microservices, combined with containerization and cloud are what make it possible to manage scale tightly.

Extensible. Because microservices are minimal AND complete, taking advantage of microservices-based solutions should not require customization, or even a complex level of configuration. If a microservice requires a lot of configuration, that would suggest that it is not the most minimal set of functionality possible.

Think of a process flow, usually depicted as multiple steps assembled in a logical order. A monolithic solution provides the whole process, and the connections between the steps are hard-coded. If you don't like the order of the steps, or you have a special need that is not provided in the solution already, then you typically have to customize the solution – develop a set of code that is not supported by the vendor – in order to get the process the business needs.

Microservices are more like the individual steps of a business process, one service for each step in the process. If you don't like the steps, then don't deploy the ones you don't need. If you need a custom step, write your own microservice to add in. Rearrange the process through orchestration, rather than through customization. This ensures that if any step of the process should change, the microservice that enables that step can be evolved without impacting any of the other steps. And if the process changes significantly, none of the capabilities are lost, but the process itself can be re-orchestrated to more closely match the new demands.

Mobile First. Recognizing that more and more users (whether employees, customers, or even business partners) are interacting with mobile devices, Aptos felt it was important to design user experiences that can be delivered on small screens, on a resource-constrained device that may or may not have good connectivity. In that context, Aptos also recognizes that users on mobile devices are often task-focused and/or easily distracted, and has taken a user experience approach that enables this kind of use.

But mobile-first design also becomes the foundation for separating the user interface from the rest of the technology stack. Even mobile phones (which have been slowly getting larger over time) will not be the smallest screens forever, as smart watches and wearables move deeper into the market for both consumers and employees. Taking a mobile-based approach, and one that separates the user interface from the functionality that is delivered to the user, is important for maintaining flexibility.

Store Resilience. There is no point in providing a solution that does not enable stores. Whatever the future of stores, they will still be an important part of the retail industry. And they will still be diverse enough and far-flung enough that an always-online assumption for store-based technology is a very high-risk proposition.

Solving the store resilience problem, while leveraging cloud-native solutions, is hard. For Aptos, it has been a patent-pending effort. Our approach was to provide disconnected selling operations, with an over-arching goal of never losing a transaction, anywhere. Several components of the retail platform services within the Aptos ONE platform provide the local data sync and local management of transactions, to ensure that every transaction is captured and ultimately communicated to the enterprise.

Store resilience then becomes the foundation for other "local" challenges, too, like on-the road district managers or merchandise buyers, who may need access to capabilities and data, even in environments where an online connection is not guaranteed. By solving the store resiliency problem, Aptos has unlocked potentially any user from the confines of the enterprise, while maintaining data consistency and accuracy, and security.

Modern Technology Stack. Microservices architecture is evolving, and the technology solutions to enable that architecture add new capabilities at a pace that almost outstrips consumer technology adoption. For Aptos, it was important to leverage the latest tools and techniques without becoming overly dependent on them. For example, while Aptos ONE originally leveraged Amazon Web Services, the platform was designed in a way to explicitly prevent dependency on AWS. There are many unique features within AWS that would make it easier to manage the platform, but our development approach took the time and effort to achieve the same management features, without relying explicitly on AWS. The proof of that approach came when it became necessary to move the platform onto Microsoft Azure: the initial work to move to Azure took approximately three weeks – less than two development sprint cycles for the Aptos team.

Modern DevOps. Moving to a microservices architecture requires organizing and managing a development team differently than in a monolithic environment. It requires thinking differently, too. One of the most important differences is in development capacity. The old adage that "you can't have nine women make a baby in one month" (a common refrain in monolithic environments where the dependencies between functionalities are very high), is actually not true with microservices. Provided you have done your homework in achieving minimal AND complete services, development becomes horizontally scalable, because every team can work independently of every other team. This is what Amazon means when they refer to "two-pizza teams". A development team of 6-10 people, working on a small, designated set of microservices (or potentially even one single microservice), is their rough rule of thumb in helping to determine if they are working on something small enough, but complete enough to stand alone.

Modern DevOps, evolving an agile development approach, also enable continuous improvement of services. All work should be defined small enough to fit within 1-2 week development cycles – with functioning, deployable code always the objective at the end of each sprint. This means the ability to deliver innovation and new capabilities to Aptos clients faster than ever before possible, without creating complexities or unforeseen consequences because of the hidden dependencies often found within monolithic applications.

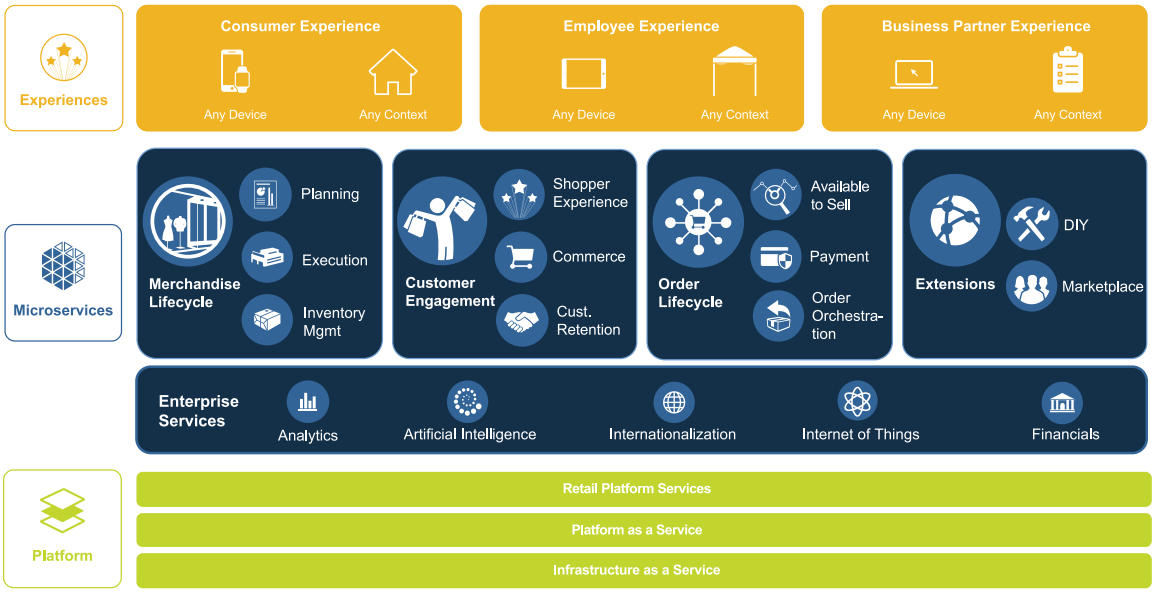
Solving the Hardest Problems First

There have been many attempts at providing cloud-native point of sale. But until Aptos ONE, none of them enabled local store resiliency within a cloud-native architecture, without relying on server hardware in the store. Many solutions have taken the risk of offering online-only capabilities in the store, but every retail CIO knows that their tenure can easily live or die by their company's ability to take the sale. In this age of exceedingly high consumer expectations, "being down" in a store is just not an option.

Even as Aptos ONE tackled that challenge head-on, enabling local deployments without store servers, but with the resiliency to never lose a transaction, we also discovered that we had a unique opportunity to reimagine commerce, given all of the current and evolving omni-channel demands that retailers face. Rather than simply porting existing solutions to the cloud, Aptos has taken the approach of evaluating whether the "old way" of doing things is really the omnichannel way. As retailers struggle to manage point of sale transactions (single point in time transactions) alongside online orders (long life transactions with many updates along the way), taking a truly "commerce" and channel-agnostic approach has become an essential part of Aptos' journey to microservices. Aptos ONE is not just cloud-native. It is omnichannel-native.

Aptos ONE

Aptos ONE is architected around three major areas of functionality: the platform, microservices domains, and the user experience.



Platform

At its core, Aptos ONE is a platform. It is based on a host of open technologies, to manage cloud delivery, as well as Infrastructure as a Service. Additionally, Aptos has developed proprietary retail services, which assist in managing the local data synchronization and transaction resiliency that makes it possible to operate cloud-native applications in a store, without a store server.

Microservices Domains

Core to a microservices architecture is the concept of "domain-driven design". The concept embodies the principle of minimal AND complete, by grouping business process capabilities together according to context, and by working to ensure that this bounded context is the smallest possible common denominator of meaning among capabilities.

Aptos defines four "parent domains", with an additional parent-domain placeholder for microservices that are developed by third parties (retailers, systems integrators, etc.) as extensions.

Customer Engagement

Within the Customer Engagement parent domain, Aptos defines three main domains for development focus: Shopper Experience, Commerce, and Customer Retention. Each of these domains encompass a major stage of the customer journey in retail: pre-transaction activity, the transaction itself, and post-transaction activity. Shopper Experience and Customer Retention map to Aptos CRM, as well as additional customer engagement capabilities in other Aptos Retail Suite solutions. Commerce capabilities map to Aptos Store, Aptos Digital Commerce, and additional commerce capabilities in other Aptos Retail Suite solutions. As of this writing, Aptos has currently developed microservices within the Commerce domain only.

Merchandise Lifecycle

Within the Merchandise Lifecycle super-domain, Aptos defines three main domains for development focus: Planning, Execution, and Inventory Management. The Planning domain loosely maps to the TXT Retail Solutions, Execution maps to Aptos Merchandising, and Inventory Management maps across all of the Aptos solutions that touch on inventory management.

Order Lifecycle

Within the Order Lifecycle parent domain, Aptos also defines three main domains for development focus: Available to Sell, Payment, and Order Orchestration. Most of these capabilities map to Aptos EOM, with some crossover into Aptos Store, AOM, and Digital Commerce.

Enterprise Services

The Enterprise Services parent domain houses all of the capabilities that cross each of the other parent domains. For example, any services for the delivery of capabilities related to Internet of Things, Analytics, or Artificial Intelligence will be managed from within the Enterprise Services super-domain.

User Experiences

Aptos focuses on three main user experiences: consumer experiences, employee experiences, and business partner experiences. Within each of those experiences, there is a focus on user context, captured as both the device in use and the location where the device is being used. Consumer context might involve a mobile device on the go, or a kiosk presented in a store. Employee context might involve a mobile device in a pop-up location outside of the retailer's real estate or network. Business partner context might involve a tablet inside a retailer's store.

The objective is to develop user experiences completely independently of the functionality delivered. Especially for consumer experiences, the user context is constantly evolving. Today's smartphone may be tomorrow's smartwatch or car dashboard or in-game virtual reality interface. By separating user experiences from the microservices delivering those experiences, Aptos aims to ensure retailers' access to flexibility and innovation in the future.

Moving to Aptos ONE

In order for retailers to be positioned to take advantage of Aptos ONE, they must first embrace many of the organizational changes that make a microservices architecture successful. A technology organization used to customizing code, building point-to-point integrations, and making large, complex upgrades every 2-5 years is not going to be well-positioned to take advantage of the flexibility and speed of the continuous delivery model needed to support microservices. It's not just an issue of skill sets. It's an issue of culture and mindset.

Cloud Foundations

Most retailers exist at one of these three levels of maturity when it comes to microservices:

Beginner. These retailers primarily operate licensed software applications within their four walls. Some applications may even be so heavily customized that they are "off maintenance" from the original software provider. Where the company has made cloud-based software investments, they have been driven by business needs and budget, and have been done outside of the approval or management of the IT department.

These retailers may be feeling the pain of brittle legacy systems that are difficult and expensive to change or maintain, but they have not yet embraced modern architectural principles for how to create speed and flexibility within their IT solutions. Before they can truly take advantage of the benefits of the Aptos ONE platform, they would do well to move to a cloud-hosted environment first, so that they can one, transition their IT organization away from support and toward innovation, and two, start to build the skill sets necessary to successfully manage applications delivered via the cloud.

Competent. Retailers competent at cloud management have all of their solutions hosted in the cloud, have developed the skill sets for managing cloud-delivered solutions, and are ready to take advantage of more cloud-native capabilities. The next step in their mastery of cloud-native is to move to a continuous delivery IT organization, otherwise known as one following modern DevOps practices. This includes implementing Agile programming methods, automation of testing and deployment functions, and moving to much more frequent software updates – with all of the change management and business training implications that this requires.

Once a retailer has moved to modern DevOps practices, the organization is ready to start down the road of microservices. This means experimenting with microservices design, including building and deploying microservices-based solutions. For some retailers, this has meant experimenting with things like voice interactions (building Alexa or Google Home skills), or with creating connections to external services like Instacart or Uber for home deliveries.

Mastery. Retailers who have developed experience with microservices, who have implemented the organizational, technology, cultural and mindset changes needed to create a modern DevOps IT organization, and who are dedicated to continuous delivery models for enhancing business capabilities over time are retailers who are ready to embrace the opportunity provided by Aptos ONE.

Commerce First

Aptos ONE has taken the approach of solving the hardest problem first: the offline resiliency required to be successful in stores. But focusing on commerce first has another advantage for retailers: it's pointless to support consumers through a shopper journey, only to get them to the end and say "sorry, I can't take that sale" – often resulting in an employee pointing a customer to a line or to a trip home to "just buy it online".

The transaction is the most important thing in retail. It's not the only thing, and it's not the only important thing, but nothing else matters if the retailer cannot take the sale. As the business becomes more omnichannel over time, retailers are feeling this pressure more and more. You can provide store employees with tools to help them sell, but it doesn't matter how excellent the shopper journey is if it falls apart because the employee can't make a transaction because of the barriers thrown up by existing systems.

Replacing store functionality is difficult and expensive. Most retailers have settled on a strategy of "surrounding POS" by looking to implement mobile point of sale on employee-facing devices, with the aim of using the mobile platform to ultimately replace the existing POS. However, those implementations have been hampered by several factors: the challenges of operating iOS or Android devices in a Windows environment, the offline/resiliency challenges of trying to work with a mobile device within a system designed for local store servers, and the high cost of security within POS when operating in a mobile environment. But even more important, most mobile solutions deployed in stores today were never designed to be truly mobile. They may provide a client for accessing functionality on a store server, but with the constant round-trip calls between client and store server host, the speed and resilience of these deployments suffer – and so does the customer experience.

Aptos ONE's commerce services enable retailers to achieve all of the goals of their mobile POS strategy, while addressing all of the challenges. It is difficult to absorb how much Aptos ONE commerce services change the game until you unbox a brand new, untouched Apple iPod, authenticate it to a network (cellular or WiFi), download a store selling app, and within minutes start taking transactions.

As of this writing, Aptos ONE commerce services are best positioned for lane-busting or basic store selling. Part of the advantage of microservices is that this will change rapidly in a short amount of time. Retailers can choose to implement the core functionality of Aptos ONE in a mobile store format now, or can leverage Aptos ONE store selling for off-premise transactions, for example in pop-up stores. Aptos encourages you to stay up to date with the current progress and roadmap of Aptos ONE, as it will evolve quickly.

Over time, the capabilities available through Aptos ONE will grow, sometimes enhancing the Store 6 solution, and sometimes evolving directly out of the Store 6 solution. Additionally, Aptos ONE, over time, will be the vehicle for integrating commerce services to EOM, and from there to the rest of the Retail Suite. Retailers will choose for themselves, based on their own unique circumstances, when Aptos ONE commerce services are sufficient to merit investment alongside Store 6 or other store solution, and then again longer-term, when those commerce services are mature enough to warrant the eventual cutover to Aptos ONE.

Beyond Commerce

Aptos ONE is not the only way for a company to achieve a transition to a microservices architecture within the Aptos Retail Suite. Aptos Enterprise Order Management and Aptos Digital Commerce are both solutions that already also have begun their journey to a microservices architecture, and just as retailers are looking to transform their IT delivery organizations, Aptos is down the path of transforming its own development teams. Every product within the Aptos Retail Suite will leverage a microservices architecture more and more over time – for new enhancements, and for evolution of existing functionality. As this evolution occurs, the Aptos ONE solution designation will expand to include the microservices delivered across all of the Retail Suite domains, beyond just commerce.

Making the Journey Together

Understanding the Aptos approach to microservices in general and Aptos ONE specifically is key to bringing microservices into your enterprise. But it's not enough to ensure that your own microservices strategy will be a success. In addition to the development and architectural modernization that Aptos is bringing to its Singular Retail solutions, we are bringing together the full range of capabilities to assist retailers in their own transformation journeys.

Aptos Retail Transformation. For retailers who don't know where to begin, or who struggle to prioritize competing priorities among the executive team, Aptos provides services focused on enabling retail transformation, as the first step to undertaking a transformation strategy.

Aptos Cloud. For retailers who are at the beginner level of cloud management capabilities, Aptos provides a set of solutions and services to help retailers migrate their Aptos Singular Retail solutions into the cloud.

Aptos ONE. For retailers who are ready to gain the advantages of a microservices architecture for commerce services, Aptos ONE is the place to start.

Aptos Labs. For retailers who are interested in innovation and accelerating their ability to innovate, engaging with Aptos Labs provide opportunities to experiment with platform-driven innovation – providing all of the benefits of experimentation without giving up a path back into the enterprise for when experiments prove successful.

About Aptos

Aptos: Engaging Customers Differently

In an era of virtually limitless choice, sustained competitive advantage only comes to retailers who engage customers differently—by truly understanding who they are, what they want and why they buy. At Aptos, we too, believe that engaging customers differently is critical to our success. We are committed to a deep understanding of each of our clients, to fulfilling their needs with the retail industry's most comprehensive omni-channel solutions, and to fostering long-term relationships built on tangible value and trust. More than 500 retail brands rely upon our Singular Commerce platform to deliver every shopper a personalized, empowered and seamless experience… no matter when, where or how they shop.

Learn More

More information on Aptos is available at: info@aptos.com and www.aptos.com.

aptos™
Engaging Customers Differently.