





9 steps to build a better device fleet


When building, upgrading, or optimizing a dedicated device fleet, there are numerous considerations. We've broken this down into nine key steps that help guide you along the way — everything from defining your business model to choosing an operating system.

Step 1: Defining a business model for your devices

Deciding on a business model for your dedicated devices is a crucial first step, as it will define how you move forward with your fleet — and any potential limitations associated with it. The first thing you need to ask yourself is “Do I want recurring revenue?” Here’s what you need to know:

 **Sell and done:** The sell and done model offers the shortest turn around time, but at the cost of recurring revenue. There is no maintenance or update schedule — many non-tech, non-connected products use this model. Think about analog watches here — the product you start with is the same product you’ll have through the lifespan. This is not a good model for a modern tech company.


 **The hybrid model:** The decision to continue supporting a product after it leaves your front door with the option to upsell new features as they’re introduced and generate recurring revenue. A router that offers advanced monitoring features for a monthly fee is a good example here — it’s perfectly usable on its own, but this enhanced feature is an upsell that provides regularly recurring revenue.


 **The ARR-driven model:** This model essentially guarantees recurring revenue, as it’s part of the package. Hardware is often sold at break even price or even a loss in exchange for a monthly subscription — think exercise equipment that is essentially useless without the monthly service.





Step 2: Selecting an operating system

Once you've decided on a business model, it's time to start thinking about the operating system your devices will run. There's a lot to unpack here, but here's the gist:

 **Hardware cost plays a role here:** If you decide you want to use iOS for your device fleet, you're also committing to Apple's hardware ecosystem. iOS isn't available on its own, so you can't add it to existing hardware.

 **Free doesn't always mean free:** Linux is a compelling option for many because it's freely available, but the overhead costs of building your own software solution can be quite high.

 **A PC-based solution will be limiting:** If you want any part of your device fleet to be portable, a Windows-based solution will be challenging as it isn't specifically designed for portability or touch-based interfaces. Pair that with an added expense for simply using the operating system, and things can get pricey quickly.


 **The Goldilocks solution offers versatility and expandability:** There's a middle ground between cost, flexibility, portability, and adaptability — it's Android. AOSP Android is freely available to download and modify but simpler than Linux, runs on a wide variety of hardware unlike iOS, and is built for touchscreen devices unlike Windows.


Android isn't perfect, but we think it's the best choice for device fleets, especially if you plan on adding a variety of device types as you scale.



Step 3: AOSP vs. GMS Android

With the signs pointing toward Android as the best choice for a dedicated device fleet, there's another big decision: AOSP or GMS?

 **AOSP:** AOSP stands for Android Open Source Project. AOSP is free to download, modify, and redistribute, so it's perfect for custom dedicated devices. However, it doesn't come with a dedicated app store or many other features commonly thought of as "Android." You'll also have to build it for your devices, which requires time and money (and a dev team).

 **GMS:** This is Google Mobile Services and is a package of proprietary software and services provided by Google, like the Play Store, Gmail, Chrome browser, YouTube, and more. GMS Android cannot be modified or redistributed, so it's less useful for creating custom device experiences.

The decision to go AOSP vs. GMS is a tough one — on one hand (AOSP), you can create exactly the experience you want, but you'll have to provide the hardware. On the other (GMS), hardware is readily available in off the shelf devices, but you'll have to live within GMS' limitations. GMS hardware also gets outdated much quicker than AOSP devices, where you can manage your own update schedule.



Step 4: Off the shelf vs. custom hardware

This decision goes hand-in-hand with the previous one. If you decide to go with GMS, you'll be limited to off the shelf hardware (unless you want to go through Google's testing for GMS certification, which isn't really worthwhile in a dedicated device scenario). But if you want to go with AOSP, you'll have to decide just how custom you want your custom hardware to be.



Buying off the shelf makes sourcing easier: You can order dozens (or more) of an off the shelf device and have it quickly. The

downside is that now you're limited to security practices of the manufacturer and won't have the ability to mandate OS update and security patch schedules. Unless you find an off the shelf AOSP device (they're fairly rare), you'll also be limited by GMS constraints.



Custom hardware costs more upfront, but offers full

customization: you decide to go custom, there are two options.

First, you could work with a hardware vendor to buy readily available hardware without an operating system. You'll be restricted to what the vendor has available. Otherwise, you can go full custom — you define the design and the hardware and have a manufacturer build it. This option is far more costly.

Ultimately, you have to decide how custom you want the experience to be — can you get by with “stock” hardware and a custom OS? Or do you need a fully custom device? If your devices will require access to GMS, both of those questions are answered for you.



Step 5: x86 vs ARM

When deciding on whether to go for a custom AOSP or standard GMS experience, you'll also need to consider processor architecture — do you want x86 or ARM? Most Android devices are ARM, but custom x86 devices are an option.



x86 offers more processing power at the cost of battery life and portability: x86 processors like those provided by Intel and AMD

aren't often considered for mobile devices, but if you're running desktop POS systems, powerful kiosks, or digital signage, it's not a bad option. This is also a route to consider if you have existing Windows hardware that you'd like to update to Android (which is not only an option, but a very compelling one).



ARM is far more common on mobile devices: If smartphones, tablets, or mPOS (mobile point of sale) are your thing, then ARM


is much more efficient — especially if these devices will regularly use battery power. The “downside” is that ARM processors aren't typically as powerful as x86, so heavy applications may suffer as a result. In many dedicated device scenarios, this isn't really an issue.

The GMS vs AOSP discussion has to intersect here, as well — if you're looking to flip older Windows x86 hardware to Android, for example, you'll have to go with AOSP.



Step 6: Building vs buying a device infrastructure

So, you have your devices and Android build picked out. Great! The decisions aren't over yet — now you need a way to manage those devices. You have two key choices here: do you build your own device infrastructure or buy one?

 **Building creates a lot of flexibility, but also a lot of overhead:** If you want a fully customizable management solution, building your own is the way to go. But like with anything custom, it'll cost you — both up front and in the long run. Getting everything up and running is also very time consuming.

 **Buying is simpler, though more limiting:** If you want to get started quickly, going with an already-available solution is the way to go. You'll lose a bit of the customization of building your own (how much will depend on your device management partner), but the tradeoffs are nearly always worth it. Finding a flexible solution will be key here to get the most bang for your buck.

While it seems outlandish to consider building your own management platform, it's not unheard of. The cons here typically outweigh the pros, however, as it is truly only necessary for the most custom of solutions. Buying a device management solution will always be more economical, both long term and short.

Step 7: The app deployment dilemma

Apps for dedicated devices aren't the same as apps for consumer devices. Your app defines the core experience you want to present to your users or customers, where delivering a pleasant, memorable experience is crucial.



App delivery is more complicated than you may realize: When it comes to delivering apps to your dedicated devices, you have choices. Are you using GMS? You could deliver and update through Managed Google Play. Are you using an AOSP-based build? You can deliver and update apps locally or over the cloud, the latter of which will rely heavily on your management partner.



Updates are important (and also complex): How you deliver app updates — both the frequency and how widely available you want the update to be — will be an important decision for the health and security of your device fleet. You don't want to push the update out to thousands of devices at once, so having a small testbed is a good idea. You'll want to roll it out slowly after that to watch for issues. Can your device management provider do that?


A good app experience, both for your customers/users and IT department, is one of the most important considerations when building or upgrading a device fleet. Get it right, and your life will be much easier. Get it wrong, and, well....you see where this is going.



Step 8: Choosing an operational model that makes sense

Ah, yes, operational model. Wait, why are we talking about operational model in a guide about building a hardware fleet? Because it's important to define how your devices will create value.

 **Designing your solution:** The first phase of choosing your operational model is designing the solution. All of the things we've covered up to this point come into play here — the device type(s) you'll use, the hardware, infrastructure, app deployment considerations, and more. You need to build, design, and test your solution before moving forward.

 **Deploying and provisioning devices could be costly:** How do you plan to deploy and provision devices? Will you hand-deliver every device and set them up manually? That's costly and time consuming. Or hire a 3PL (third party logistics) company to manage the delivery and provisioning process for you? This saves time, but it's costly. You could ship devices to their intended location and provision onsite to save money, but the setup requires a significant time investment. Can your device management provider handle remote deployment and automatic provisioning? This is oftentimes the best solution for cost effectiveness and time efficiency.

There are a lot of moving parts and considerations to the operational model, and this is just the tip of the iceberg. Every step of the process — designing, building, staging, deploying, and provisioning — should be considered before you start moving.



Step 9: Bringing it all together with cloud tools and automation

So, that's a lot. Once you have your device fleet up and running, it's time to start actually managing it. You want the overhead here to be as minimal as possible...but how do you do that?



Automation from end to end saves time and money: At Esper, we believe in DevOps philosophies, but we extend that well past software delivery all the way to the edge — your devices. When you can automate normally manual processes (and do so with precision and accuracy), you save time, money, and, oftentimes, trouble.



The cloud is your friend: In order to utilize powerful automation, you need to utilize the cloud. This is where you manage devices whether they're across the room or across the country, so why not take it a step further and use the advantages presented here to deploy software and updates, configurations, and more? In fact, why not use advanced device grouping to automate the whole process?

The journey to full fleet automation, remote provisioning, touchless deployments, and the like takes time. But we know how to get you there. Get started with Esper today and start managing your fleet with more agility than ever before.





Ready for the next step? Check out our advanced fleet planner guidebook — it covers advanced concepts like app delivery models, the benefits of full device fleet automation, and a whole lot more. With these two guides, you'll have everything you need to build, deploy, and optimize a device fleet — and the knowledge to get it right the first time.

[Learn More](#)



esper.io