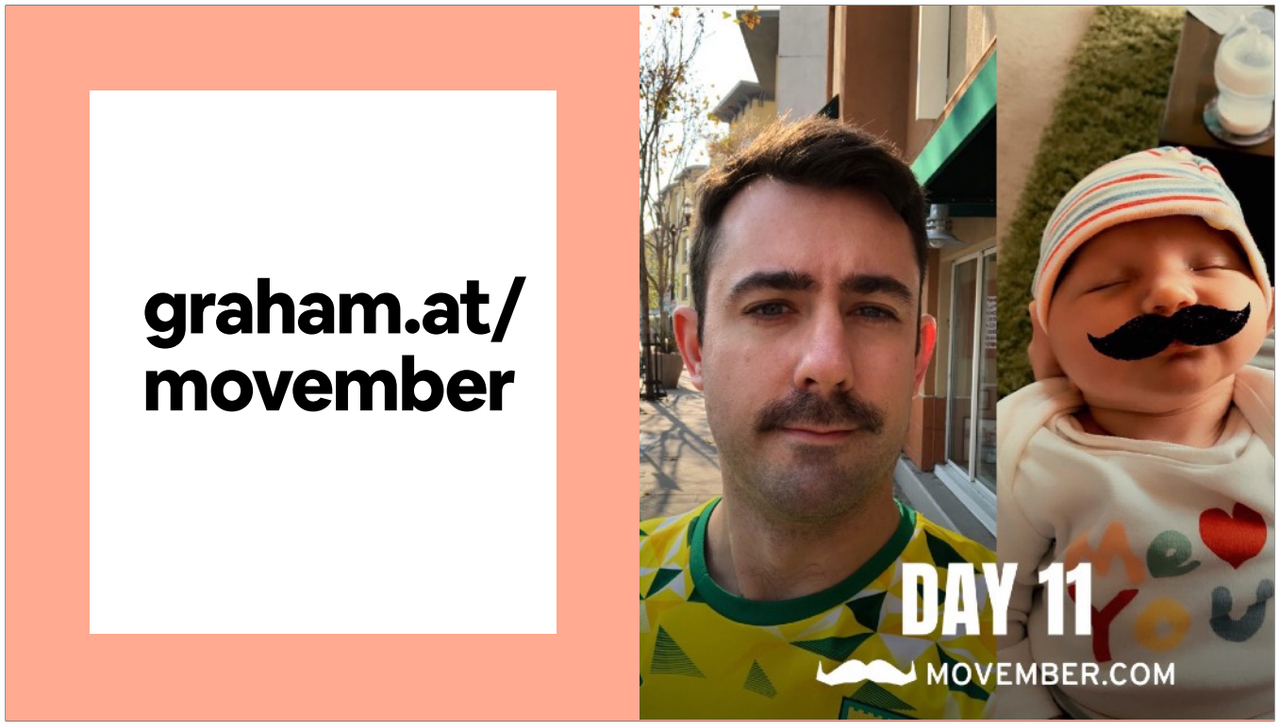
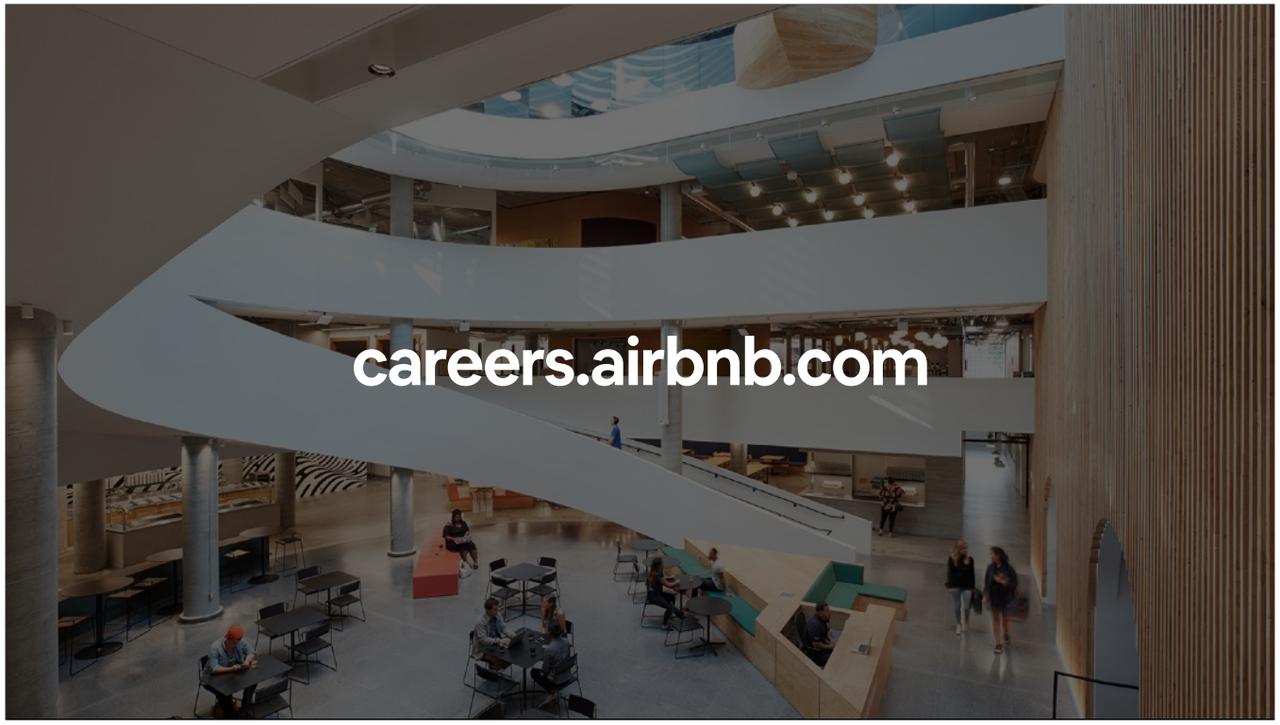
GRAHAM Good morning. My name is Graham Gilbert, and I am joined by my esteemed colleague, Brett Demetris. We are part of the Client Engineering team at Airbnb and today we are going to talk about our adventures with MicroMDM. Before we get started, I would like to show you a quick two minute video

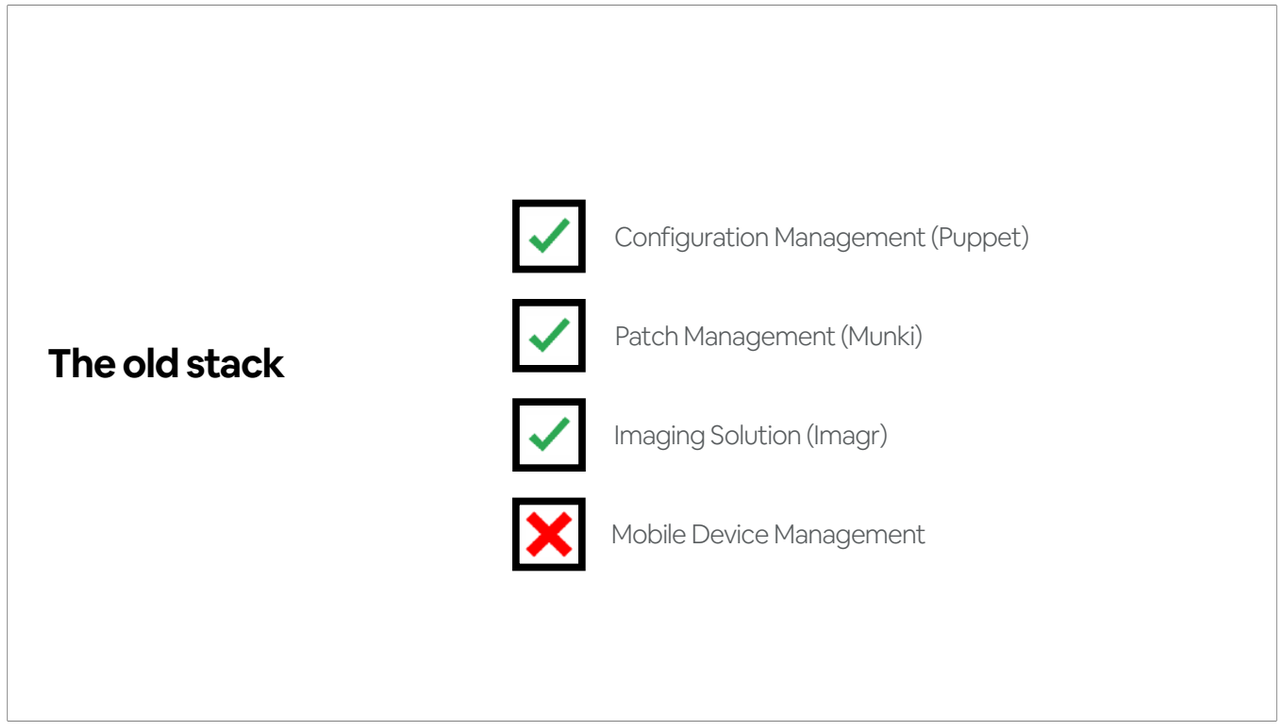
GRAHAM.AT/MOVEMBER



I am one of those one out of every 200 people. I have been in remission for about 18 months now. If you find this talk useful in any way, please find your way to graham.at/movember



And we are hiring. We have several spots open on our team, so if the madness we are going to talk to about sounds interesting, come and say hi.
And obviously the most important part of this picture is the beer taps :)



BRETT

The first 2 items aren't terribly important, but the an imaging solution does provide us some context for the start of this talk, because that's how we provisioned machines - one of the things that an MDM would (or should) do for us via DEP.

What about kernel extensions?

At this point imagr was still alive
It seemed more challenging to change our deployment process than to implement an MDM
We did need to account for the kernel extension issue
Run a package that copies in the kext policy database

Then 10.13.2 came along

We all know what happened here... (change slide)

User Approved MDM

user approved MDM and user approved kernel extension loading. Our time of faking it was at an end.
Now we were in a Rush to get mdm implemented, Insane quotes from vendors. They know what is coming, they know why you're calling them, they know you aren't ready, and it is clear who has the upper hand monetarily.

How about MicroMDM?

IT'S FREE RIGHT?

So we looked at MicroMDM. Looked like a good idea. We could get it running quickly, its just a go binary, easy to build, runs on nix and darwin. Didn't need to get approval for a large purchase from our head of purchasing. {click} Additionally, we would need to go through our vendor approval process, which added a bit more fuel to our fire.
Don't want to mess with this guy!

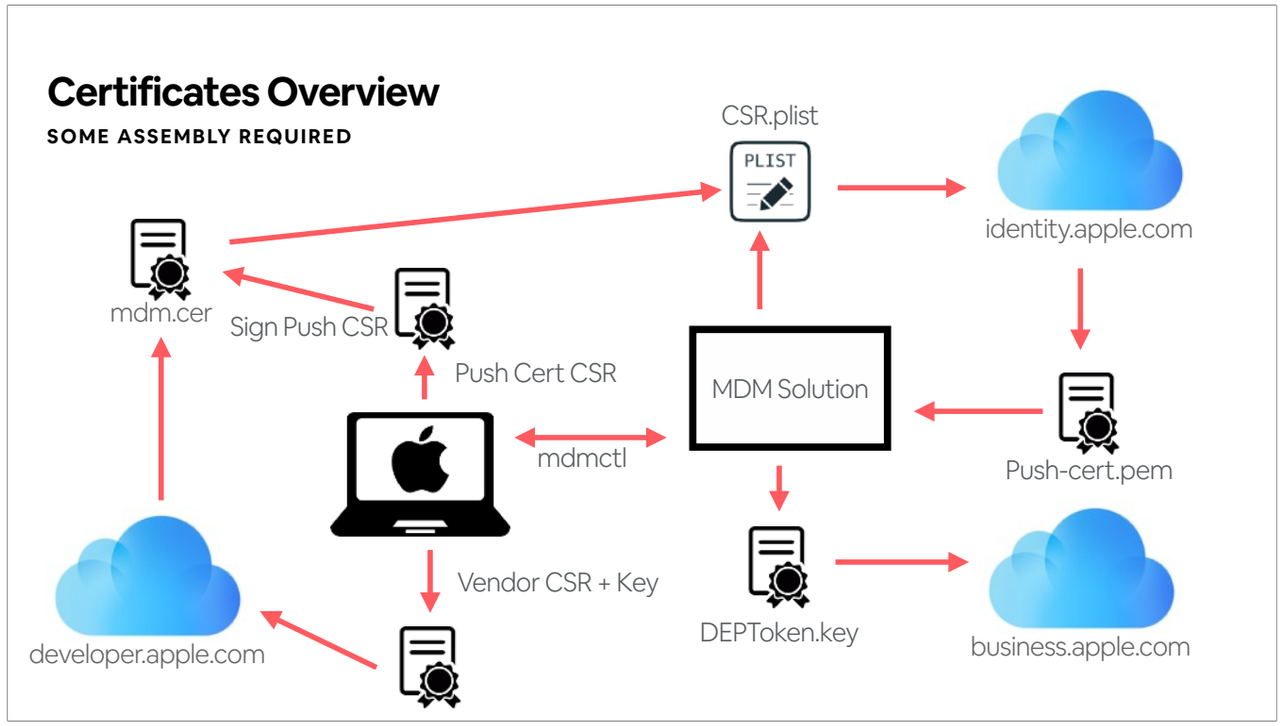
Indeed, “how hard can it be?” We asked ourselves

Step 1:

Becoming an MDM Vendor

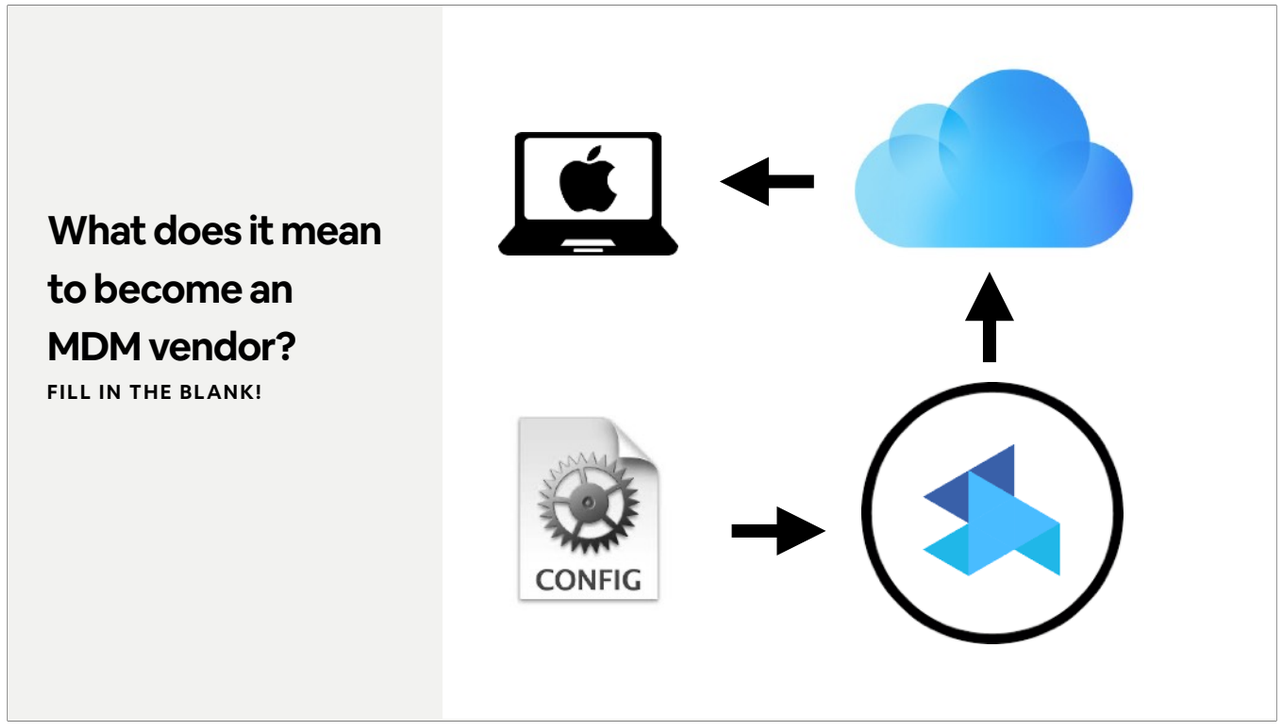
- [x] Find the right person on your Enterprise Developer account.
- [x] Speak to one or many people at Apple.
- [x] Get your vendor certs.
- [ ] Bask in glory.

MDM vendor certs are not turned on in apple developer portal by default, so we asked apple to turn it on for us.



Your mdm solution generates you a plist that contains some CSR information, you log into identity.apple.com, create yourself a certificate, upload your plist and then download your certificate which you then upload to your mdm solution - job is done at this point under normal circumstances.

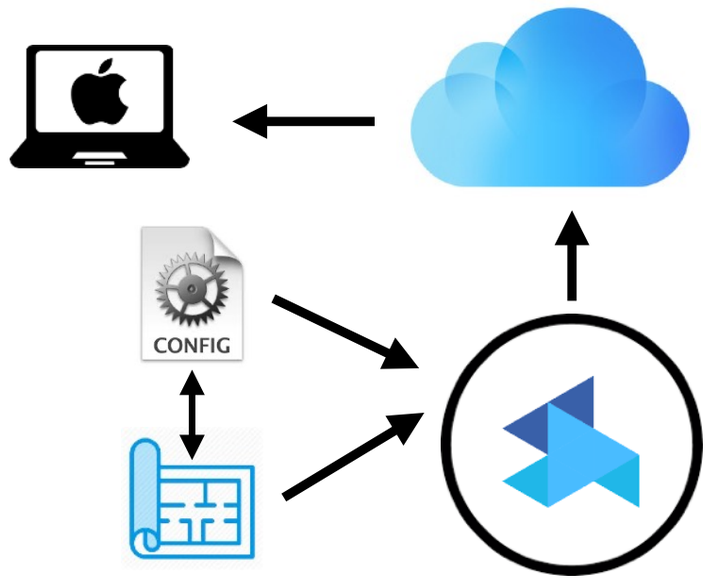
None of this is terribly complicated due to mdmctl being a great automation for certificate setup, and ultimately it's not an impassable hurdle even if mdmctl didn't exist. The point is that this is the first time you start to see the vendor side of MDM, and it should provoke some thought. If there is just this much extra complexity to something as simple as creating certificates, what other potential issues could we expect to face? This diagram is an example of a single deployment, but what might change about this diagram if we had, for example, a dev stage prod instance of micromdm? Better not worry about it yet!



Or rather, what does it mean to be an MDM vendor?
I've never been an MDM vendor, but we know what paid products do (or rather an expectation of what they do). We put configuration profiles in, and then it goes to the device. MicroMDM can do that from what we've seen playing around with it. Lets just put MicroMDM in and we should be good to go!

Okay, let’s do this

DEP AND CHILL



The first requirement we had was to provision new devices with a kext profile (remember we were faking it, and we can’t fake it anymore), so we solved this by using microMDM’s blueprints which seemed quite effective. We created an enrollment workflow that just installed some profiles. Very easy very simple!

But What About All The Other Devices?

- Script that looped over all the devices and shot out the profile
- Repeat previous step a few times
- Hope devices come online
- Enrollment blueprint deployed profiles ~80% of the time

We know that we only have profiles provisioned on enrollment, and that if we install a profile again the user doesn't see any impact, so we just sprayed the profile to everyone. We had to repeat this many times. We basically hoped that devices were online. We also started to notice that devices were coming out of provisioning missing profiles from the blueprint which started to look like a bug, or maybe inefficiency in APNS, or maybe both? We didn't really know, and our challenges seemed to keep growing… **pass to graham**

Then we needed to push a password policy profile

GRAHAM We got a requirement to push a password policy. We of course could have done the spray and pray we did last time, but there surely must be a better way of doing this.

When all you have is a hammer...

- Sal is our reporting tool
- Sal reports on installed profiles
- Sal can run code in plugins
- I know how to post profiles to MicroMDM with Python
- Sal is in the same VPC as MicroMDM
- Sal is looking like a petty sexy hammer right now...

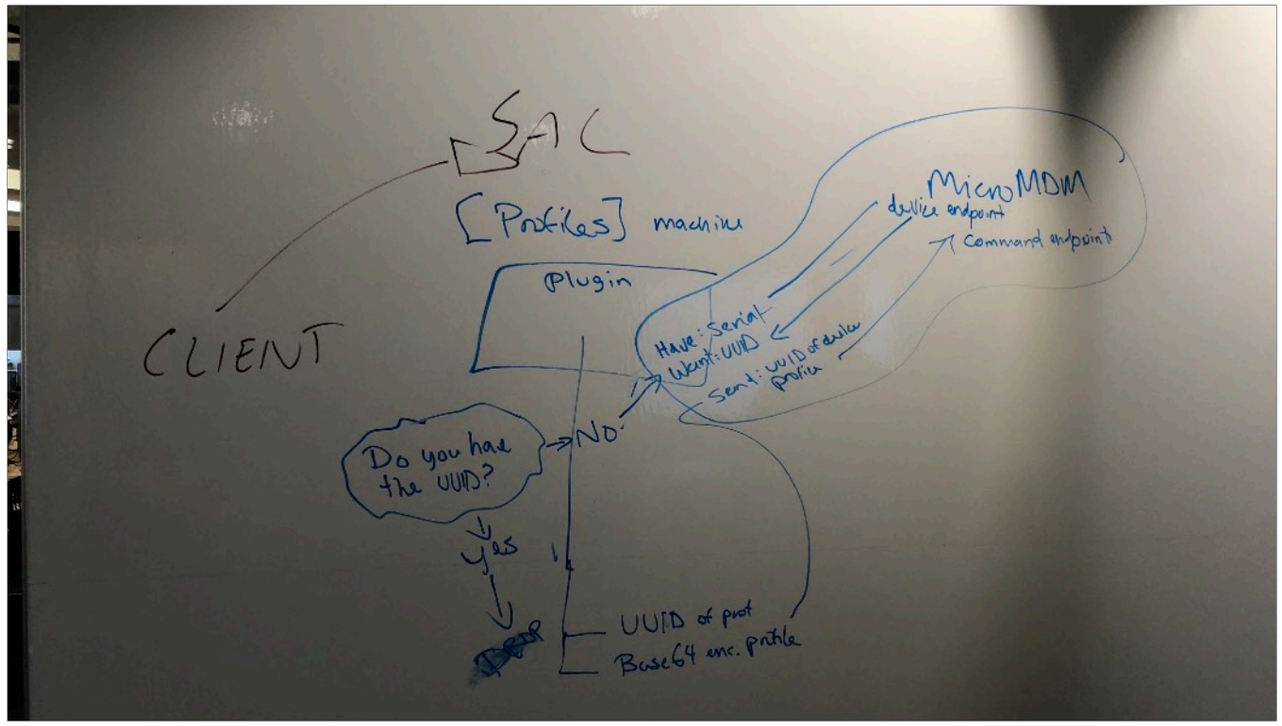
So we have this tool called Sal. {click} Sal is our endpoint reporting tool. {click} One of the things it reports on is installed profiles on macOS. {click} It also has plugins. Those plugins are written in python so we can run arbitrary code in them {click} and I know how to post profiles to MicroMDM with requests in Python {click} And it just so happens that Sal is in the same amazon VPC as MicroMDM so it's nice and easy to get them talking to each other {click} so basically Sal is looking like the worlds sexiest hammer right now

What happens if your MDM doesn’t support something?

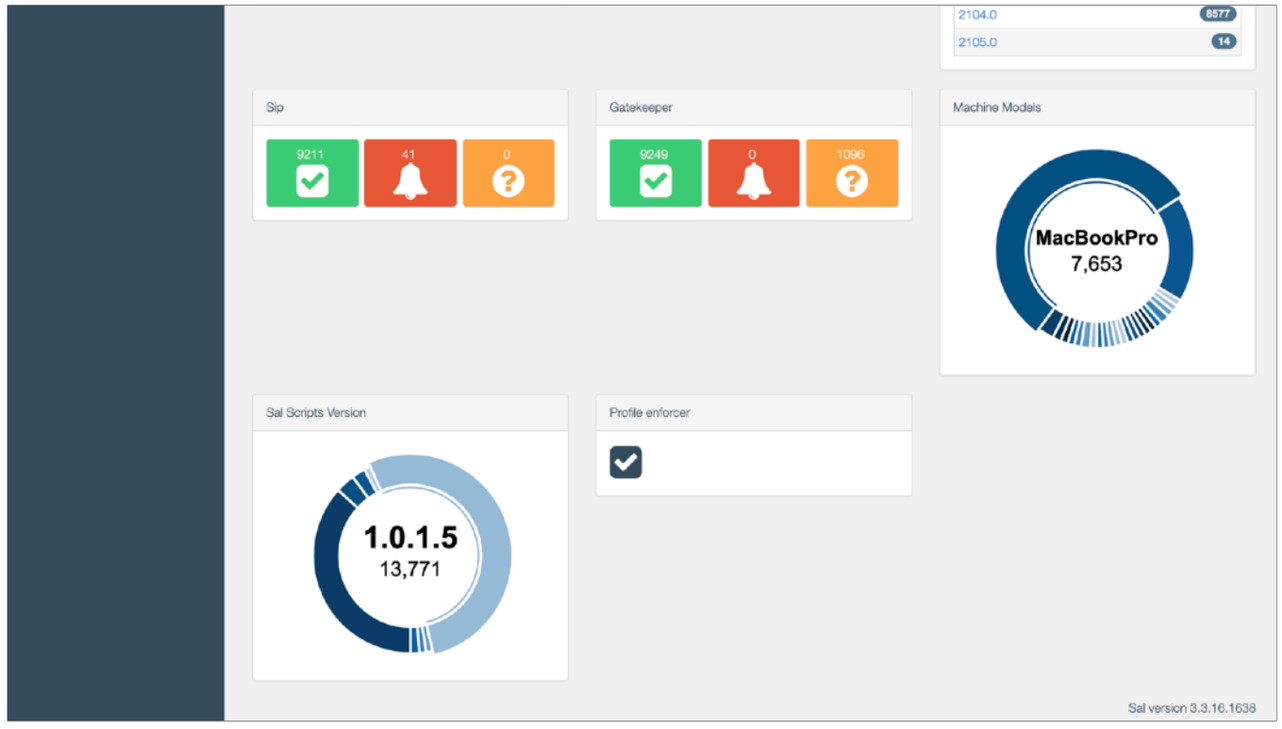
To post the profile, we needed to look up the UDID of a device from it’s serial number in MicroMDM .This wasn’t there. Normally if you need something to be added, you’re asking your vendor to do it. What happens when you’re an mdm vendor using open source mdm?

You’re doing it yourself.

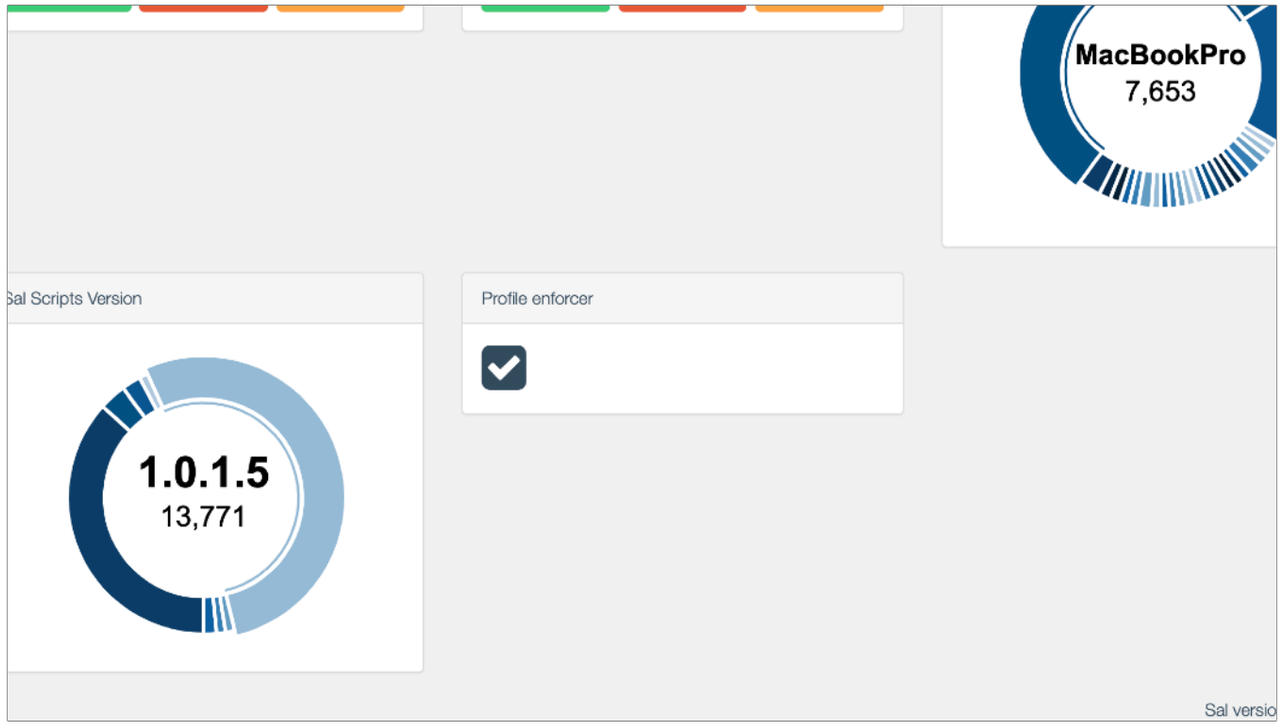
You’re adding it yourself.



So we got into the war room and did some expert planning (note the use of uuid where we really meant UDID - we really didn’t know what we were doing)

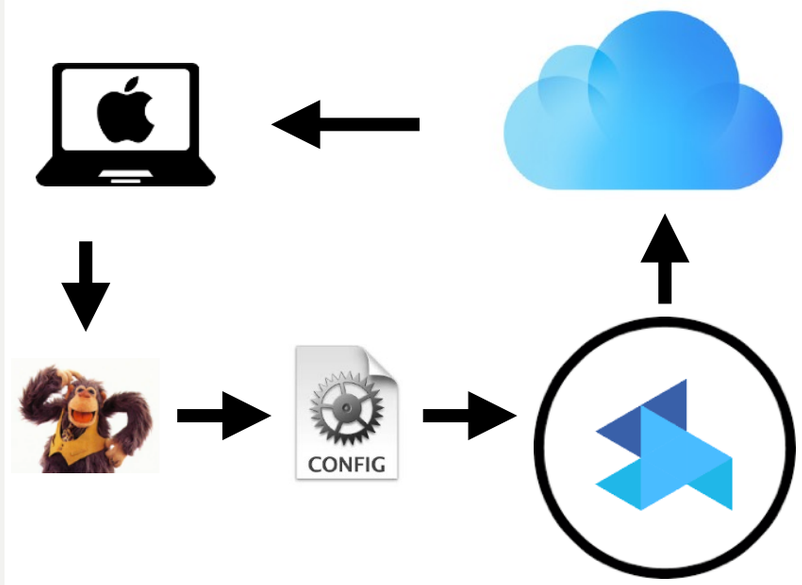

2104.0
8577
2105.0
14
Sip
9211
41
0
Gatekeeper
9249
0
1096
Machine Models
MacBookPro
7,653
Sal Scripts Version
1.0.1.5
13,771
Profile enforcer
Sal version 3.3.16.1638

{click}



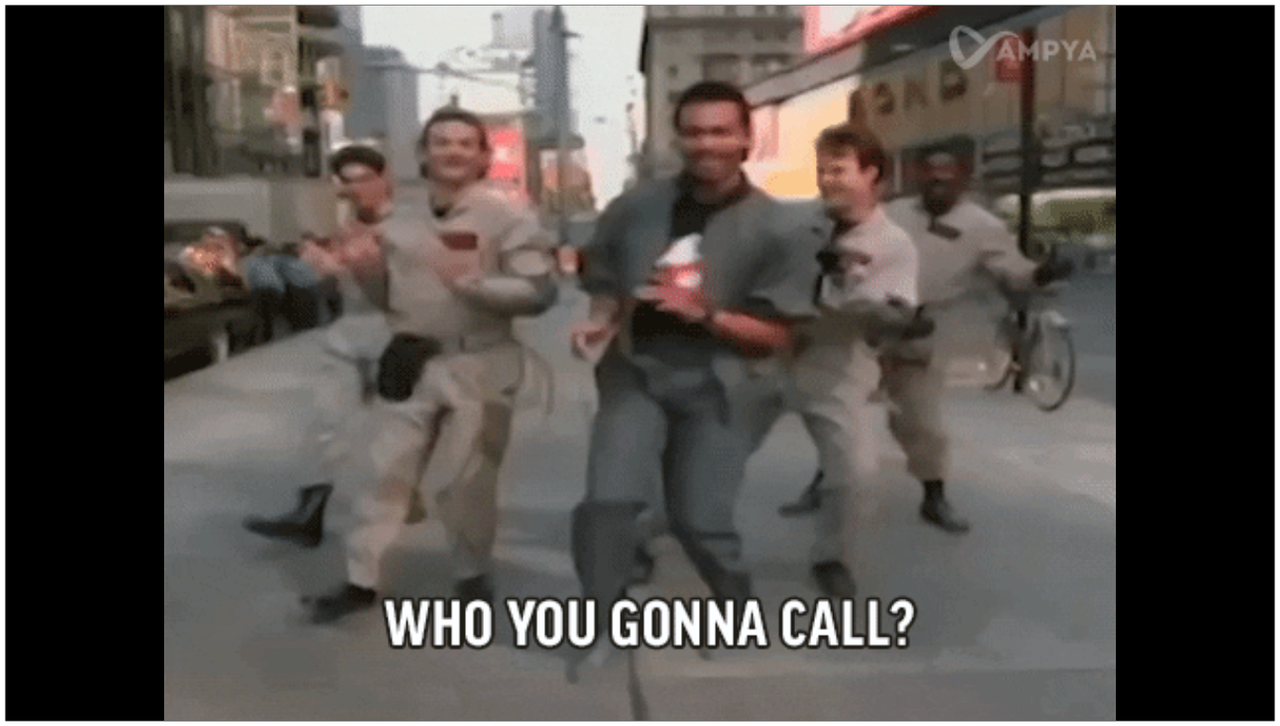
And we wrote this sal plugin in about a day and a half

Push it, push it real good

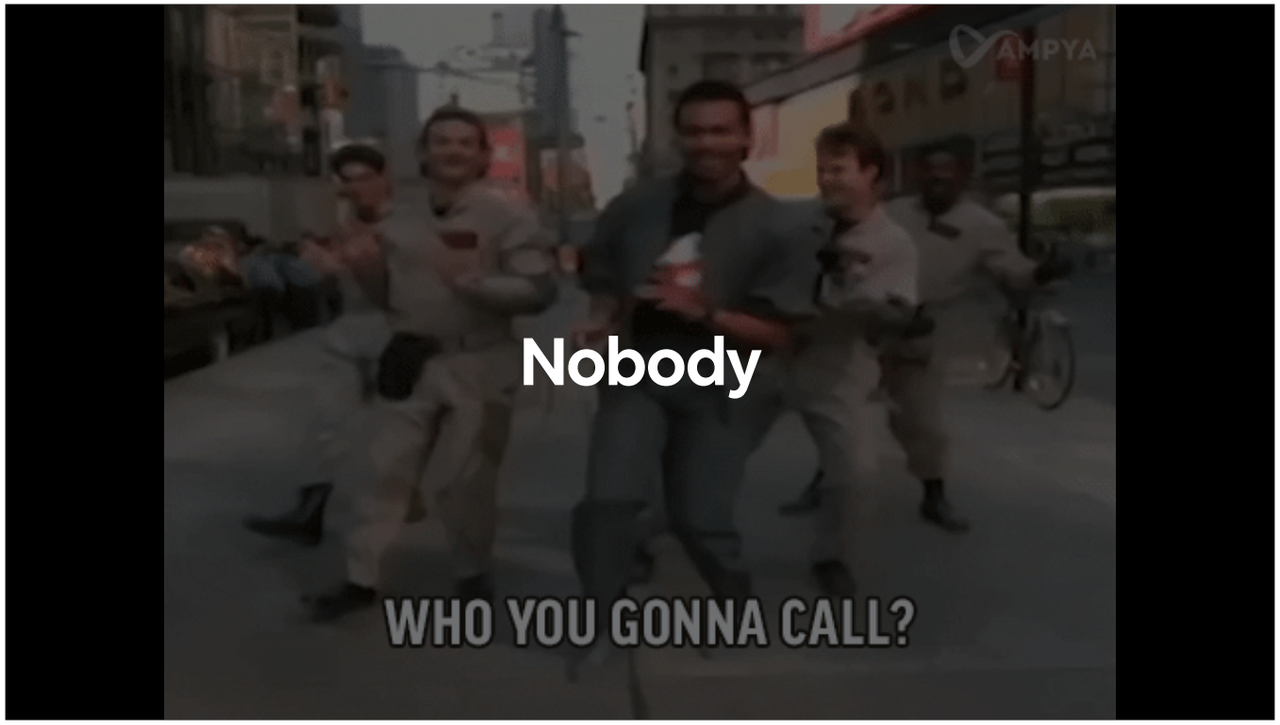


We were feeling pretty pleased with ourselves. Machines started checking into Sal, Sal saw the profile was missing and sent the installProfile command to MicroMDM.

And we sent it every 30 minutes. If APNS was having an off day (and that never happens, right?) we could be queueing up many many commands in micromdm. This caused the database to explode. We had to write some code to fix that little problem. And a fun fact: MicroMDM uses BoltDB, which stores itself in memory when it's being used. Our database grew to 18Gb, but we were running it on a box with only 16Gb. That was fun. Eventually all MDM commands ground to a stop. Our support team was unhappy. They were unable to deploy machines.



BRETT So it’s at this point most people would be calling their MDM’s support people to get help.



You’re not calling anyone (you’re the vendor, remember?)

Looking at an MDM Incident

WHAT'S IN PANDORA'S BOX?

So as graham mentioned we had an issue with micromdm's database exploding and the service going down. Just telling you that we had a problem doesn't really do it any justice. I want to take a moment and bring you deeper into our incident, and touch on the finer grain details of what went on, and how we addressed these issues. Lets open up pandora's box.

MDM Incident

DAY 0

What we Know

- Support has reported that devices are not able to pass Device Enrollment screen
  - New Hire class of ~50 people are starting in 3 business days
- MicroMDM's service is up and logging
  - MicroMDM's database is 22 GB
  - InstallProfile actions are depleted by ~90%
  - SCEP/PKI Operations are taking up to 2min to complete

Action Items

- Read up on BoltDB documentation to understand the database size issue
- Disable Profile Plugin in Sal

Our support team reports an issue with device enrollment. The severity here is pretty extreme in that when you switch to a DEP based enrollment MDM becomes your only pathway to bootstrap a Mac. There is of course an alternative, which is to bypass device enrollment by avoiding an internet connection, and manually installing our tools, but this is unacceptable for several reasons - one of which is we'd be missing our enrollment profile and any future hope of deploying profiles via MDM. This is needless to say, a horrible position to be in. We have 3 business days and our precious weekend to fix this, there really no alternative. So here is what we know...

MDM Incident

DAY 1

What we Know

- BoltDB is a key value store in written in pure golang
  - There is a compression tool built into the boltdb binary
  - There are several projects for inspecting the database
- Database Compression has no effect
  - The size of the database correctly reflects its contents

Action Items

- Review MDM Specification
- Review MicroMDM code to understand how the command queue works
- Pull a development database, and review MicroMDM's table layout

We have read the boltdb documentation and we've found a few interesting things out. The database itself is a key value store (simply a huge json file on disk), and the community has developed some tools that make interacting with the database pleasant. There is even a native command built right into the binary to conduct a compression. This looks great! The compression event takes upwards of 3 hours because of the database size. We were overjoyed to know it had no effect, the database really was a 22 gb text file - think about that! Maybe we're looking in the wrong spot. Maybe we should start to look at the micromdm code, its database schema, or possibly at the MDM spec itself to see if we can find a clue there.

MDM Incident

DAY 2

What we Know

- BoltDB runs out of memory
  - A database size greater than the server's memory is the main cause of our service interruption
- The MDM Spec lists several status codes in the results payload
  - Acknowledged, Error, CommandFormatError, Idle, NotNow
- MicroMDM's command queue is cumulative based on the MDM response status

Action Items

- Resize AWS instance for MicroMDM to something greater than 22 GB of memory
- Work on reducing the size of MicroMDM Database

Pulling the real database, and inspecting it was failing. Pulling a development database that was much smaller worked. This inconsistency flagged us to look into how BoltDB loaded itself, and sure enough it does so in memory. We now had a pathway out of our critical situation, but there is still a lot more work to do.

MDM Incident

POST MORTEM (BEER TIME)

Action Items

- Write a database tool to clean our the command queue
- Rethink our use of the Sal Plugin

Lessons Learned

- We need a firm understanding of the MicroMDM code base
- We need a complete understanding of the MDM Spec
- We should probably get better at Golang
- Being an MDM Vendor is Hard

We were having memory troubles running the bolt database tool on the server due to memory issues, so we were sort of forced to write out own that specifically targeted cleaning out the command queue. We also had to rethink how we were distributing profiles and handling machines with missing profiles.

During this outage we learned some valuable lessons. The first is if we were to continue with MicroMDM it is imperative that we know the software inside and out, we also need to know the MDM spec completely. With this is implied that you are fluent in Golang. There are no half way points here. Being an MDM vendor is hard.

And of course, MDM itself isn't a great protocol. You send a command and hope the device is online. There's no documented way to trigger an MDM checkin from the device (we know of a few from trial and error). The device doesn't even do it periodically automatically.

NotNow is my personal favorite part of the MDM spec. Did you know, that if you send a command, the device, for a variety of reasons can say “no”? Screw you mdm protocol, this is my device, you’ll do what I want you to.

As consumers of MDM you have no reason to really know about MDM responses. We learned about these the hard way. Remember the database incident we mentioned? That was a product of NotNow.

DEP is never down

Of course DEP is never down. The green light on the status page tells us so.

Most of what we need isn’t even possible with MDM

Really, most of what we need isn’t even possible with MDM. How we traditionally want to manage an endpoint is with a client, in fact in some cases many clients that ensure each other are alive and managing the device in the way our security overlords demand. Not to mention a ton of configuration we need to do on the device simply isn’t available via profile yet.

Mobile Device Management

So it’s not really mobile device management

Maybe Device Management

It’s maybe device management. But apple is forcing us to use this junk. We have no choice.

Getting Help

The best place to get help with setting up and ongoing maintenance of MicroMDM is the MacAdmins Slack. Join by getting an invitation here.

Once you join Slack, the following channels will be useful.

- `#micromdm` for MicroMDM specific questions.
- `#mdm` and `#dep` for generic questions about MDM and deployment programs.

For defects and feature requests, please open an issue.

Not a product!

MicroMDM is not a full featured device management product. The mission of MicroMDM is to enable a secure and scalable MDM deployment for Apple Devices, and expose the full set of Apple MDM commands and responses through an API. But it is more correct to think of MicroMDM as a lower level dependency for one or more products, not a solution that lives on its own.

For example, MicroMDM has no high level options for configuration profiles. It accepts an already composed `mobileconfig` file and queues it for a single device at a time. Device Management products often have built-in support for signing profiles or pushing them to a chosen device group. MicroMDM expects those features to exist in a higher level companion service.

MicroMDM has no Web UI.

MicroMDM can enable disk encryption and escrow a secret, but it has no option for storing that secret on the server.

Dynamic enrollment / user authentication workflows belong in an external service. MicroMDM serves the exact same enrollment profile at the `/mdm/enroll` server endpoint for every device. It also does not care about how the enrollment is protected from unauthorized devices. The number of possible workflows are infinite, and the recommendation is to point the devices at a separate URL for serving the enrollment profile.

As you see, MicroMDM itself lacks many features that are usually present in device management products. But it also exposes a low level API that would allow an organization to build a product that is highly custom to ones environment. Over time, the community will likely share solutions that depend on MicroMDM and expose higher level workflows.

Before using MicroMDM, consider that there are a number of alternative, commercial products like Airwatch, SimpleMDM and

We had more problems than just those created by Apple. We had plenty of our own making. We were pretty naive when deploying MicroMDM. Just just stood it up and walked away pretty much.

invitation here.

Once you join Slack, the following channels will be useful.

- `#micromdm` for MicroMDM specific questions.
- `#mdm` and `#dep` for generic questions about MDM and deployment

For defects and feature requests, please open an issue.

Not a product!

MicroMDM is not a full featured device management product. The mission
MDM deployment for Apple Devices, and expose the full set of Apple MDI
But it is more correct to think of MicroMDM as a lower level dependency f
its own.

For example, MicroMDM has no high level options for configuration profile
file and queues it for a single device at a time. Device Management produ
pushing them to a chosen device group. MicroMDM expects these featur

Of course this wasn't a good thing to do. More recently, groob posted this wiki page. MicroMDM is not a product.

~~**We needed it to be a product**~~

We needed a product

We kind of needed it to be a product. We realized that we need a product...**hand to graham**

Blueprints

- Blueprints specify actions to take when the device enrolls
- Blueprints are a one shot - no ongoing management
- If they work, great
- If not…
- Well, it’s mdm

GRAHAM. Blueprints specify actions micromdm will take when a device first enrolls. {click} The problem is that blueprints are a one time deal. There is no ongoing management of profiles out of the box. {click} Which means it’s great if they work. {click} If they don’t…. {click} Well, it’s mdm.

Let’s talk about APNS

- Fun fact: devices rarely check in on themselves
- And the schedule they do is completely undocumented
- No way to trigger a checkin from the device
- APNS is the only way we can trigger an MDM checkin
- Good job we control all of the networks our devices are used on…

Devices rarely check in by themselves. {click} And when they do check in is completely undocumented, so we can't rely on that. {click} And we can't even trigger a checkin from the device (and apple closed my feedback requesting thay as "wont fix") {click} Most mdms would use their agent to know when the device is online, but we're trying really hard to do things the way that Apple wants us to, so we want to use MDM only. So our only option at this point is to literally hammer the device with APNS requests and pray the device comes online at some point {click} and I don't know about you, but I'm really glad that I control all of the random networks that my users go on...

I heard you like updating profiles

- No versioning built into profiles
- Apple expects us to use the UUID as a version
- Or keep track what profiles have been sent

There is no versioning built into profiles, {click} and although this to my knowledge is undocumented, apple expects us to use the UUID of profiles as a version, {click} or to keep track of every single profile we send

Profiles: The gift that gives one time

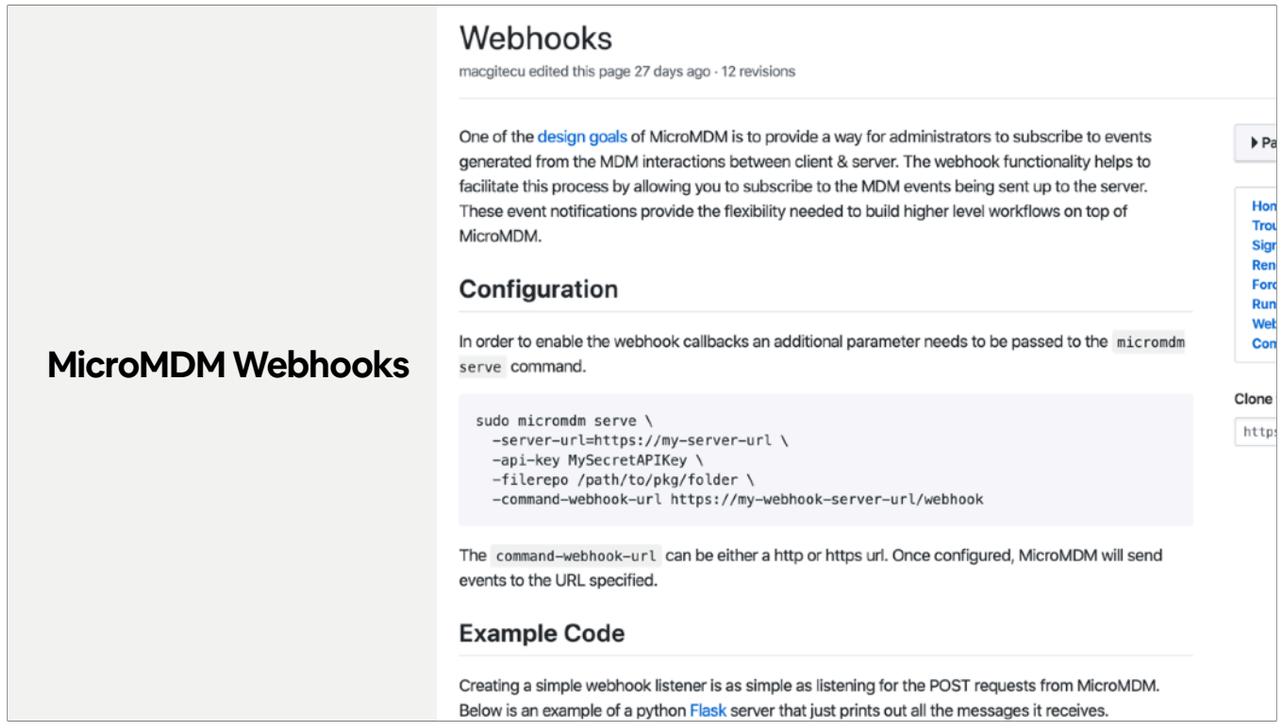
- Profile payloads are evaluated **during install only**
- Too bad if the profile payload isn't supported on the OS you're running
- Even worse if it is supported, but something else changes the setting out from under the profile

Profiles are evaluated during install only. {click }Which is irritating if the profile you need to manage isn't supported on the current running OS. {click} Or even more irritating if it's supported, and then the upgrade process removes whatever storage mechanism is underneath. Basically our only option here is to re-send EVERY SINGLE PROFILE when the build number changes.

We needed it to be a product

So we needed it to be a product.

MicroMDM Webhooks



Webhooks

macgitecu edited this page 27 days ago · 12 revisions

One of the design goals of MicroMDM is to provide a way for administrators to subscribe to events generated from the MDM interactions between client & server. The webhook functionality helps to facilitate this process by allowing you to subscribe to the MDM events being sent up to the server. These event notifications provide the flexibility needed to build higher level workflows on top of MicroMDM.

Configuration

In order to enable the webhook callbacks an additional parameter needs to be passed to the `micromdm serve` command.

```
sudo micromdm serve \
  -server-url=https://my-server-url \
  -api-key MySecretAPIKey \
  -filerepo /path/to/pkg/folder \
  -command-webhook-url https://my-webhook-server-url/webhook
```

The `command-webhook-url` can be either a http or https url. Once configured, MicroMDM will send events to the URL specified.

Example Code

Creating a simple webhook listener is as simple as listening for the POST requests from MicroMDM. Below is an example of a python Flask server that just prints out all the messages it receives.

Groob pointed me to the web hook feature of Micromdm. Basically you can specify a http endpoint and every single response that comes into micromdm will be sent to it. This started the beginnings of an idea

AN OPINIONATED MDM ORCHESTRATOR

MDMDirector

We wrote a tool we have called MDMDirector. It attempts to make MDM as stateful as possible, by regularly requesting device information and profile information and then making decisions based on the information we receive. I call it opinionated, because it is designed around our very specific workflows.

What does MDMDirector do?

- Handles initial device enrollment (InstallApplication, InstallProfile, DeviceConfigured)

We have a workflow that runs at device enrollment - MDMdirector will install all of the appropriate packages, profiles and then send the device configured command.

What does MDMDirector do?

- Handles initial device enrollment (InstallApplication, InstallProfile, DeviceConfigured)
- Profile state management

MDM director checksums each profile payload it is told to install, and will use that to work out if the profile is present (via profilelist). If the profile is missing or out of date, it will be reinstalled. If we have told MDMDirector to remove the profile, it will make sure it is not there.

What does MDMDirector do?

- Handles initial device enrollment (InstallApplication, InstallProfile, DeviceConfigured)
- Profile state management
- Dynamic profile signing

It will sign every profile before it ships it to micromdm

What does MDMDirector do?

- Handles initial device enrollment (InstallApplication, InstallProfile, DeviceConfigured)
- Profile state management
- Dynamic profile signing
- Shared Profiles and Apps

Shared profiles and apps are installed everywhere. It makes it easy to get a common baseline of profiles deployed across your fleet.

What does MDMDirector do?

- Handles initial device enrollment (InstallApplication, InstallProfile, DeviceConfigured)
- Profile state management
- Dynamic profile signing
- Shared Profiles and Apps
- Device Profiles and Apps

Device profiles and apps are deployed to a single device. This might seem silly until you consider...

What does MDMDirector do?

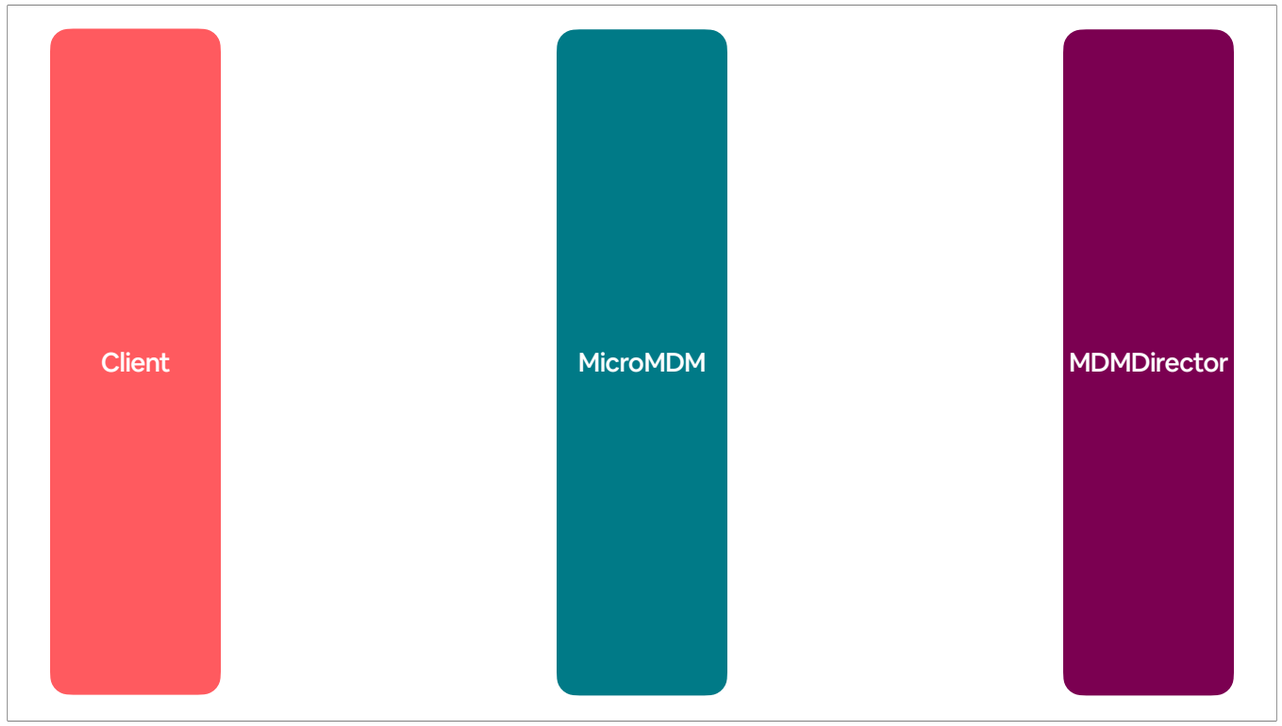
- Handles initial device enrollment (InstallApplication, InstallProfile, DeviceConfigured)
- Profile state management
- Dynamic profile signing
- Shared Profiles and Apps
- Device Profiles and Apps
- RESTful API

API: Allows us to continue to use tooling like Puppet to generate the profiles and use mdm to deliver them or CI/CD tools to manage the profiles in source control.

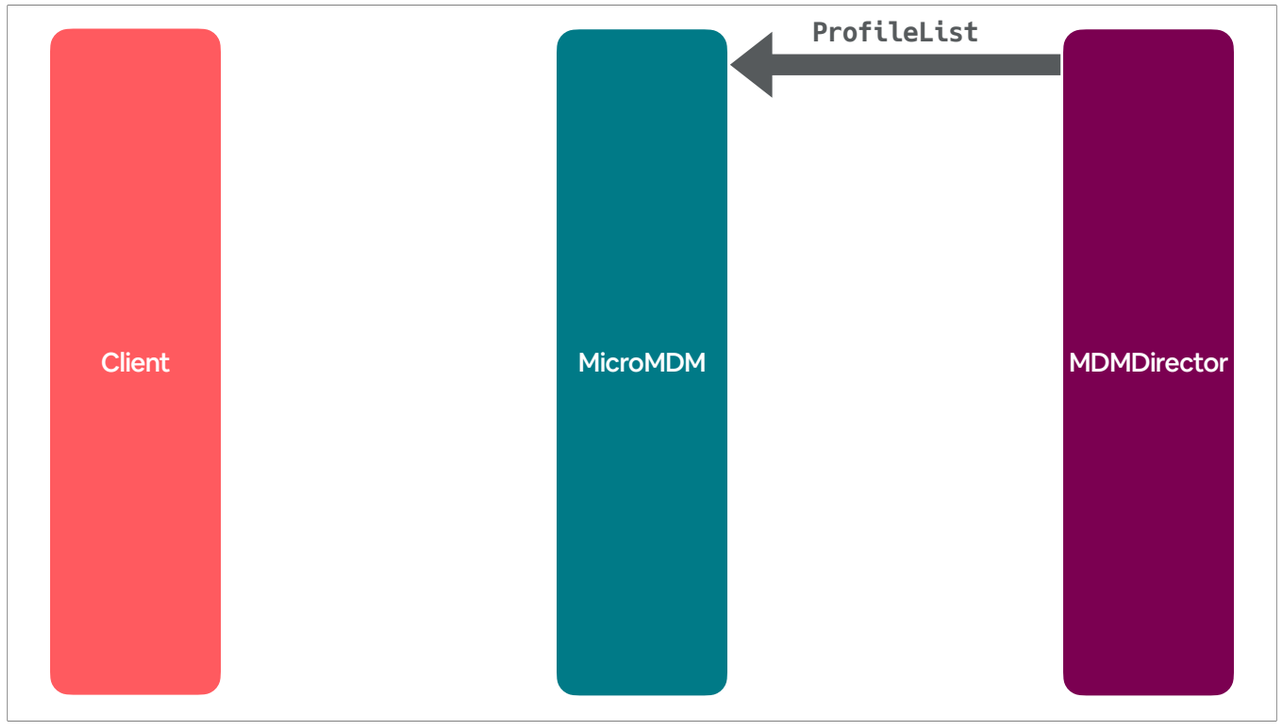
What MDMDirector doesn't do

- No GUI
- No groups
- Little flexibility out of the box

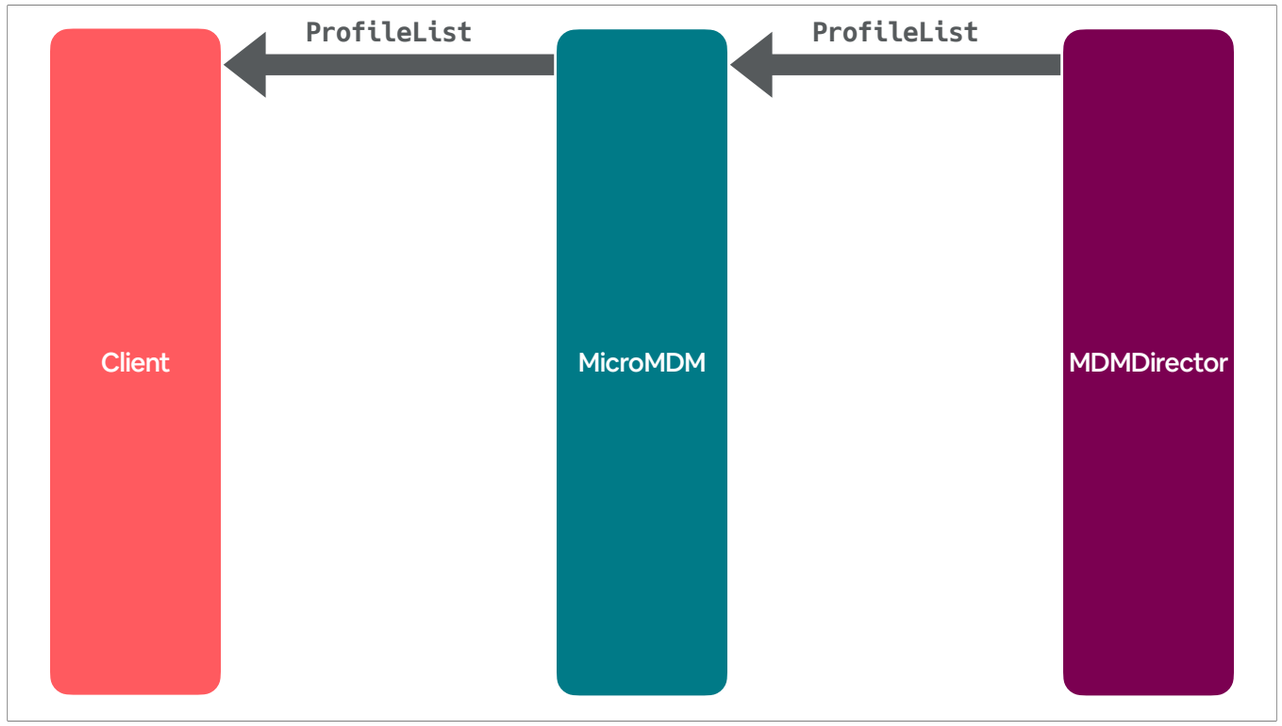
MDM Director purposely doesn't have a GUI. It is configured via an API, so there is nothing stopping an interested party in creating one. {click} It doesn't have a concept of groups - it can either push things to all devices or one device. This is because we use tools such as puppet to compose individual profiles. {click} The big one is that it is definitely opinionated. It is designed for our workflow - but with additional tooling that decides what apps and profiles go to which devices, it can be incredibly flexible.



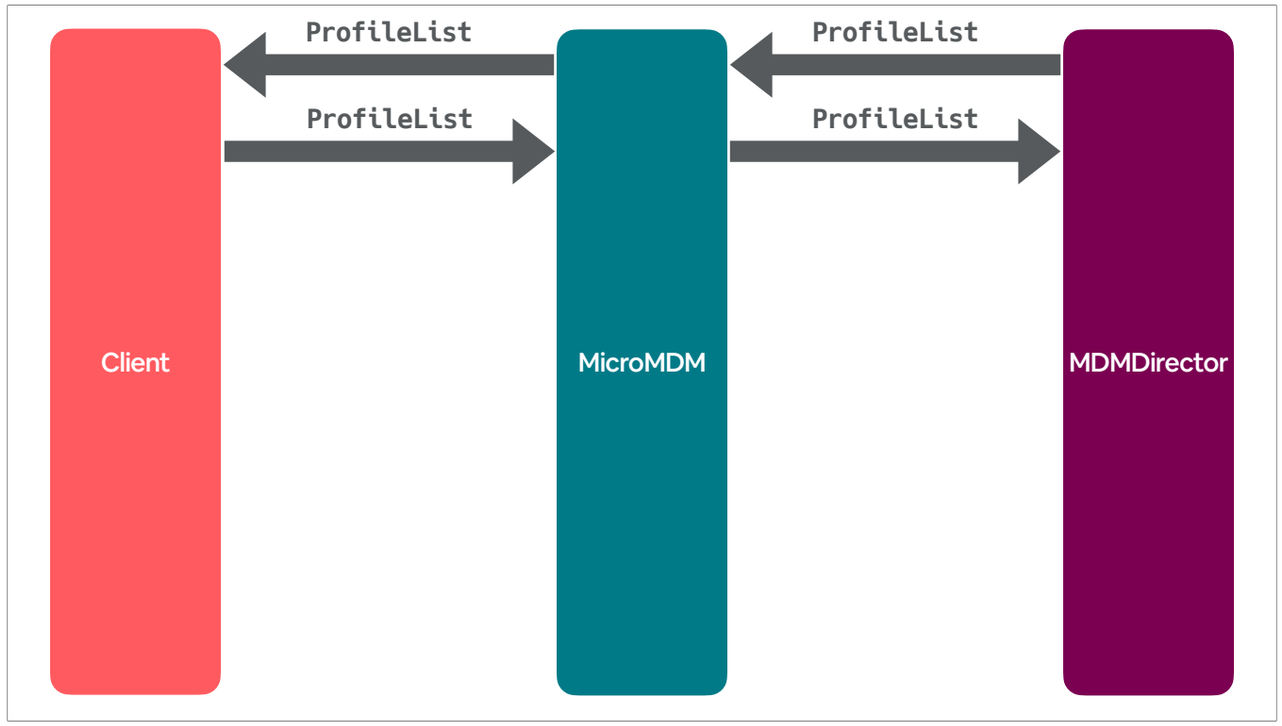
So let’s talk about profile management with MDMDirector. First off notice that MDMDirector never talks to a client directly. It is always sending it’s commands via MicroMDM - this means that MDM director doesn’t need to be accessible via the internet (and probably shouldn’t be.



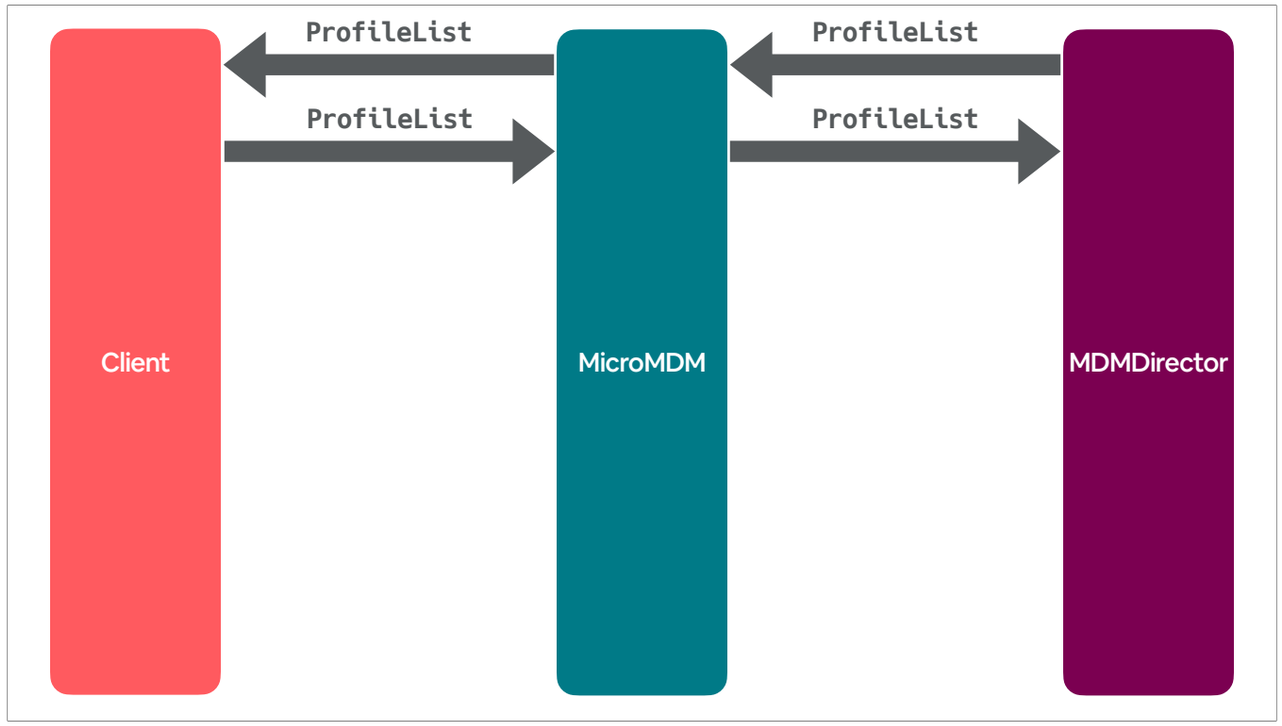
Every hour or so, if the client hasn’t spoken to MDMDirector for a while, MDMDirector will request several commands from the client. They include Device Info, Security info and the one we care about today, ProfileList



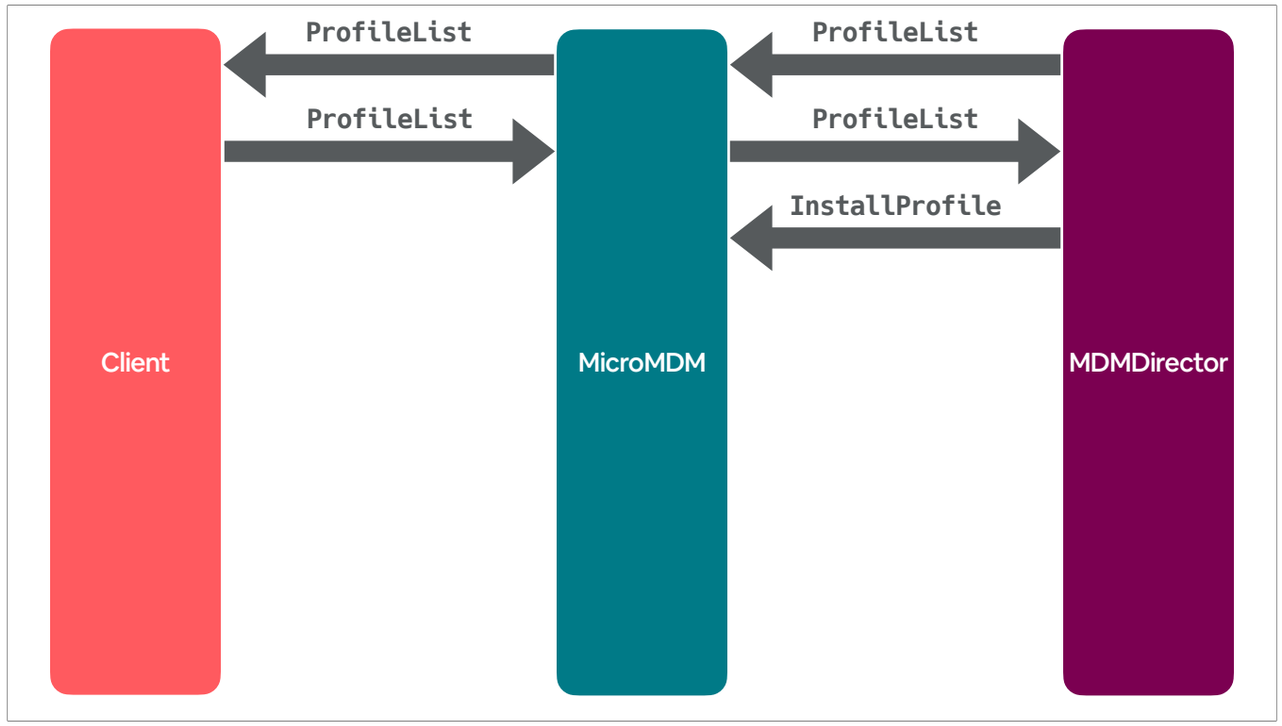
That command gets accepted by MicroMDM and it attempts to send the command to the client.



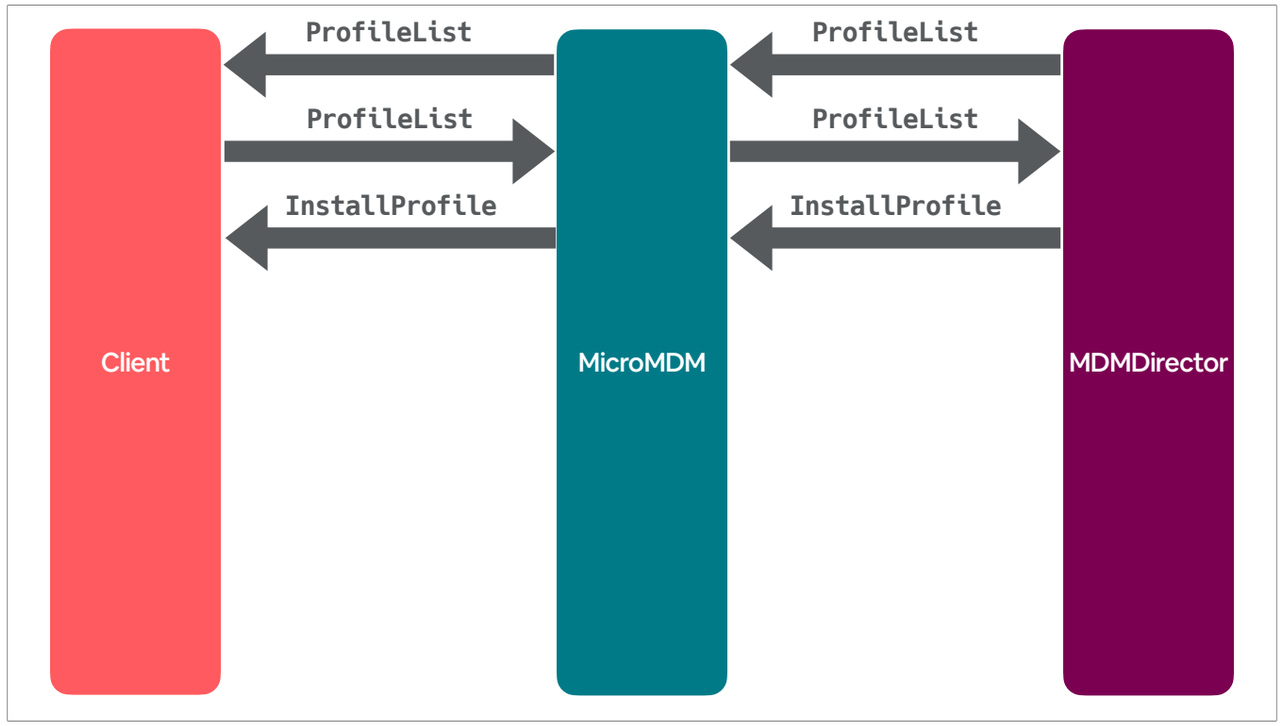
We got lucky and the device is online. It sends back the profile list and MicroMDM passes the response to MDMDirector via the web hook



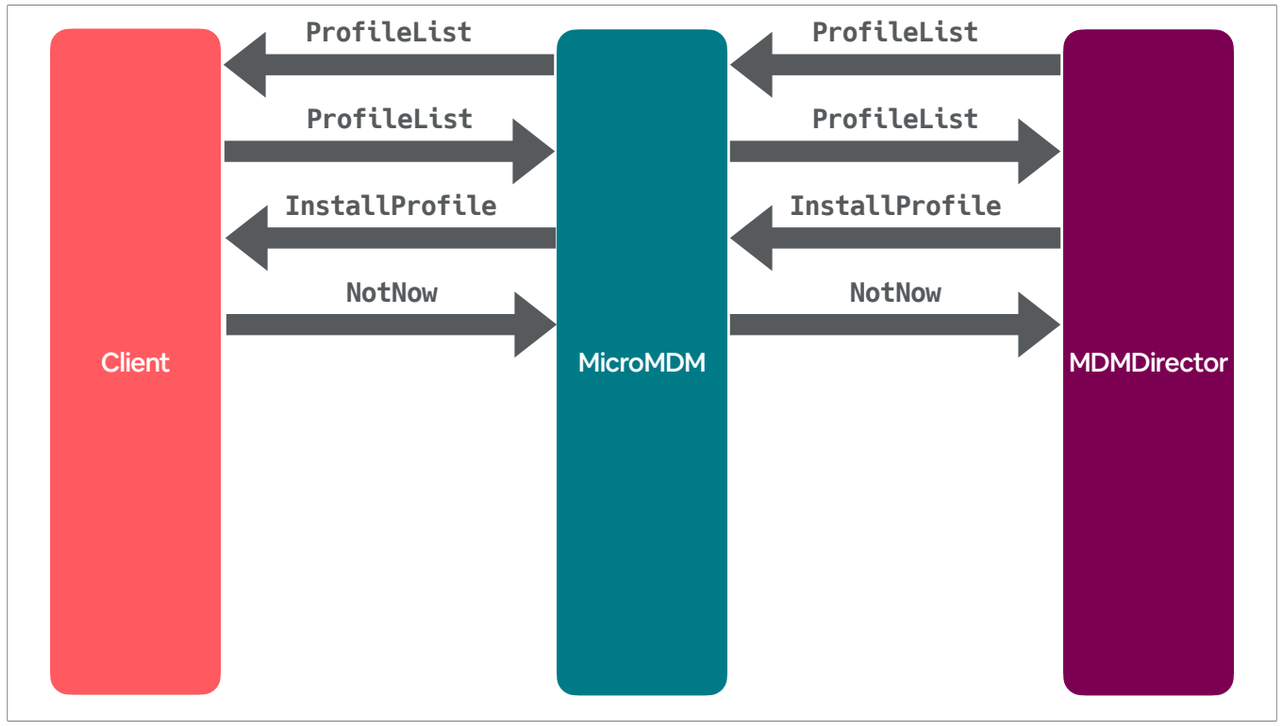
Now we are at the point of deciding what profiles should be installed. MDMDirector goes over the list of profiles that are currently on the device, and compares them to what is stored in it’s database.



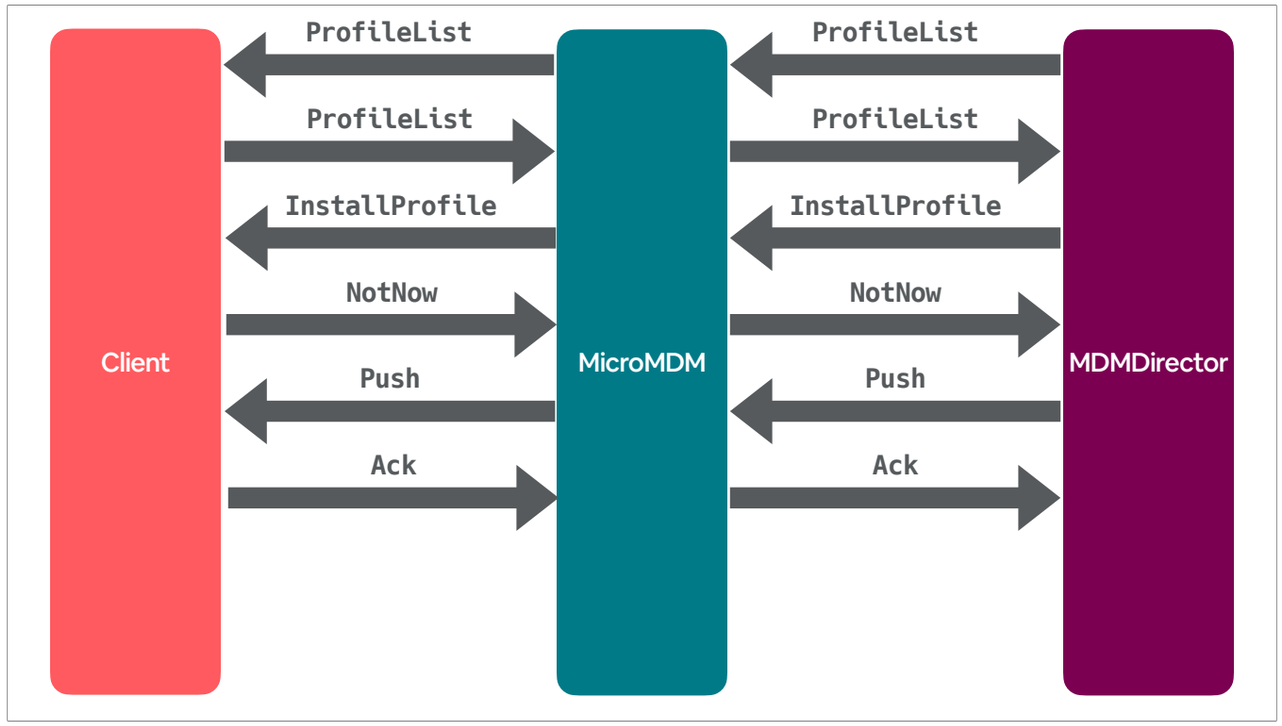
MDMDirector will hash the payload it has stored - the desired state - and will set that as the profile's uuid. If the uuid that is receives from ProfileList differs it will tell MicroMDM to install the profile. In this case one of our profiles is out of date, so we replace the uuid with our calculated hash, sign the profile on the fly and send the command over to micro



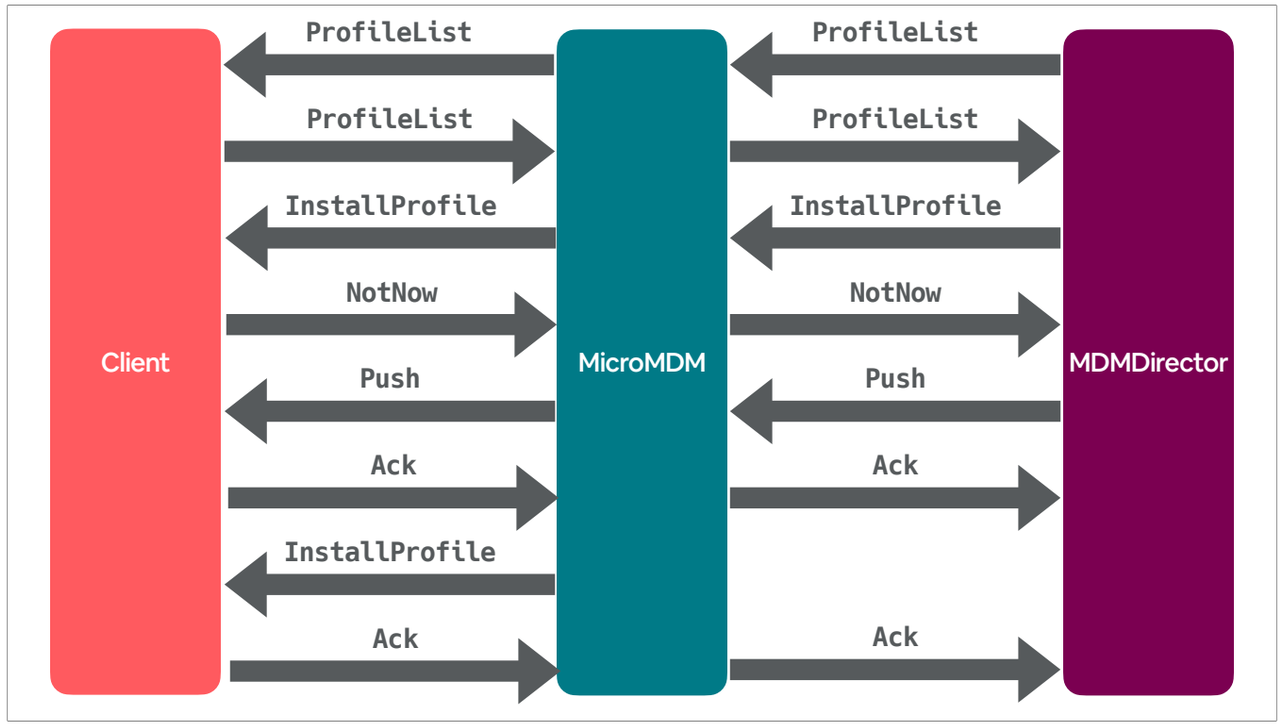
Which sends it over to the client



Uh oh. If we just had MicroMDM, this would be the end of the story. The petulant toddler wins, no profiles for this mac.



But MDMDirector doesn’t accept that kind of nonsense. It pokes the device to talk to Micro again



MicroMDM sends the command again from it’s queue and the world is happy again.

The future

- Control of more of the MDM spec (EraseDevice, DeviceLock)
- Tool to track state of applications to orchestrate InstallApplication
- Enrollment profile management (track installation and query cert expiry)

In the future we would like to implement control of more of the MDM spec - right now we are looking at methods of implementing erase device and device lock in a sensible manner. {click} At present installaplication is a spray and pray for anything outside of the app store. We are thinking of ways we can build a tool to report information back to MDMdirector in a useful manner. {click} We are also looking into ways of managing the enrollment profile - this is to renew the scep certificate (most mdm’s get around this by issuing the cert forever btw)

So at this point I hope you’re thinking about implementing MicroMDM - after all, if these idiots can do it, it can’t be that hard, right?

Probably not.

No, you probably shouldn't. I was speaking to a chap at the beer bash last night. We spoke about the difficulties he was having getting MDM vendor certs. We had a back and forth about how we got them, and then I asked him how many machines he is looking after. He answered that he has about 100 devices. I told him right away that he shouldn't implement MicroMDM. MicroMDM requires a lot of tooling to be written around it. It is often a full time job for both of us. Most people will be fine with SimpleMDM, Airwatch, Fleetsmith or any of the other fine MDM vendors. However, if you have specific requirements, such as wanting everything to be configured by code in source control, and you have people on staff who both can write an application to orchestrate MicroMDM such as MDM director, know enough Go to debug MicroMDM when it goes wrong AND has a pretty deep knowledge of the MDM spec, then writing your own mdm might be for you.

Thank you!

- We are hiring! careers.airbnb.com
- MDM is woefully inadequate, but it's all we've got
- File feedback with Apple about what is missing
- graham.at/movember
- github.com/mdmdirector/mdmdirector
- #micromdm on MacAdmins Slack

So that's it. Thank you to Patrik and the rest of the team, thank you to Tycho for listening when I asked him why there are so few European speakers all those years ago. Thank you to Victor Vranchan (or as most of you will know him as Groob) for writing MicroMDM and of course thank you to you all for making it here so early after the beer bash.