

Hello! Can I have a quick show of hands? Who's managing lab environments?

And by that, I mean computers that are shared, by multiple users?

Who's still imaging? Me too - ish.

Are you concerned about the direction Apple is taking as it seems to be dropping support for this? Me too. Who's moving, or has moved their workflows to DEP? Me too - ish.

I'd like to share the journey I'm taking - and I say TAKING on purpose, because I don't think we're quite where we want to be just yet.



Neil Martin

Senior Desktop Support Specialist
University of East London



@neilmartin83



I'm Neil and I joined UEL in 2004, working in one of its schools supporting specialist music facilities. Like lots of folks who do what we do, I didn't have an IT background - I had a music degree and I was a sound engineer.

Back then we had a couple of labs of eMacs and that's where I got bitten by the Apple bug. One thing led to another and now I'm with IT Services.

<C> I'm also honoured to be one of the administrators on the Slack, where you may know me as this person.

The University of East London

Over 19,000 students

3 campuses

470 Macs, 200 in labs

2015 - 2018, Jamf Pro

April 2018, Jamf Cloud

jamf NATION
Roadshow



UEL has more than 19,000 students, spread over our Docklands, Stratford and University Square Stratford campuses.

We manage around 5,000 PCs, with Macs making up about 10% of the estate. On both platforms, we manage 2 key sorts of experience - a 1-1 model for staff, and a shared “Lab” model for students.

We've been managing our Macs with Jamf Pro since 2015 and we've just migrated to Jamf Cloud. The migration was really smooth - we worked with Moof-IT and Jamf, who were both great - and it meant we could get rid of 4 on-premise servers, saving my team and I time as well as resources.

“

A rising star of education. The most improved university in the UK over the past decade for the quality of its student experience

- Times Higher Education Student Experience Survey

”

In this year's Times Higher Education survey for Student Experience, we were called out for increasing our score by nearly 15 points since last year, a jump that was higher than any other UK university.

Resources

<https://soundmacguy.wordpress.com/2018/05/17/lab-nauseam-dawn-of-the-dep/>



Before I begin, a copy of these slides, resources and links to all the other things I'll mention will all be on my blog, so don't worry about taking notes.

Once upon a time...



I'd like to tell you a story. Once upon a time, one summer, there was a lab.

<C> and it was a happy lab.

It was time for the Macs to get their annual re-image up to the current version of macOS.

Once upon a time...



So, you'd NetBoot the lab, remotely.

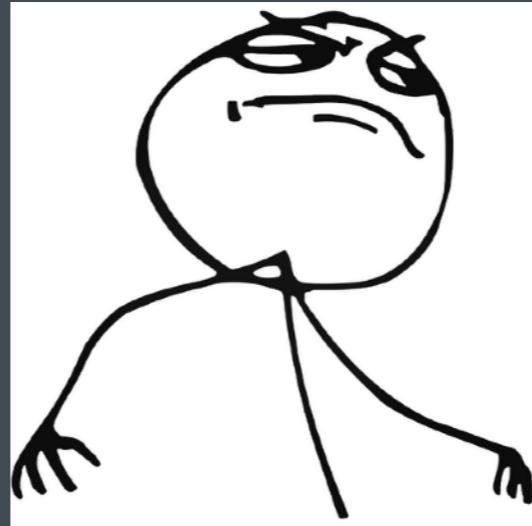
<C> Jamf (or Casper) Imaging would open, then Autorun imaging would kick off.

Once upon a time...



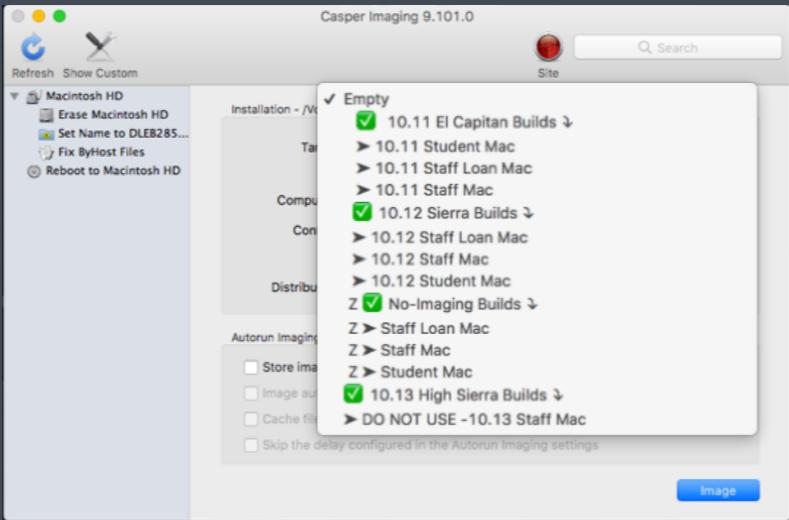
Magic!

Once upon a time...



And you'd walk away, feeling pretty good about yourself.

Once upon a time...



If you were provisioning a new Mac, you'd give it a hostname

<C>, then choose a configuration that told Jamf the version of macOS you wanted and that you wanted it to be a Student facing computer.

Its role (whether the Mac was staff or student facing), was stored in an Extension Attribute.

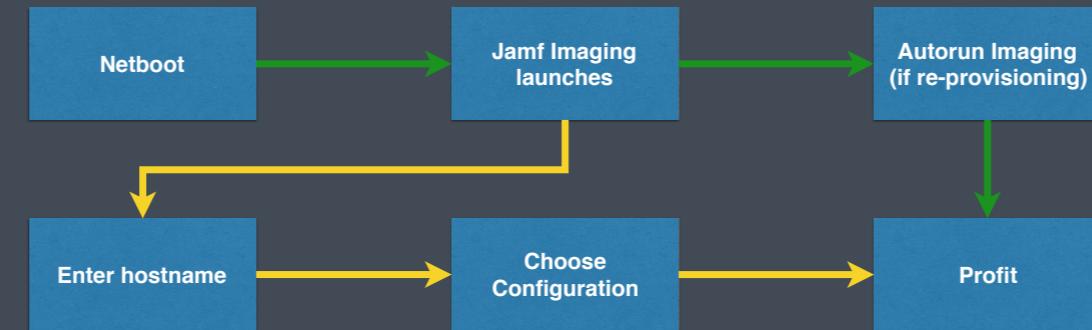
Depending on what you'd select here, staff or student, we'd write a corresponding dummy file to the Library folder and using the API, have a script set the proper value of the Extension Attribute based on that file.

Once upon a time...



After you clicked the Image button, you'd walk away, feeling pretty good about yourself.

Modular imaging workflow



This modular workflow might look like this. Once you hit that Imaging button...

Modular imaging workflow

Profit

AutoDMG - base image

MDM profiles - some preferences

First-run script - everything else

Policies - software etc

We'd lay down a clean, never-booted AutoDMG image...

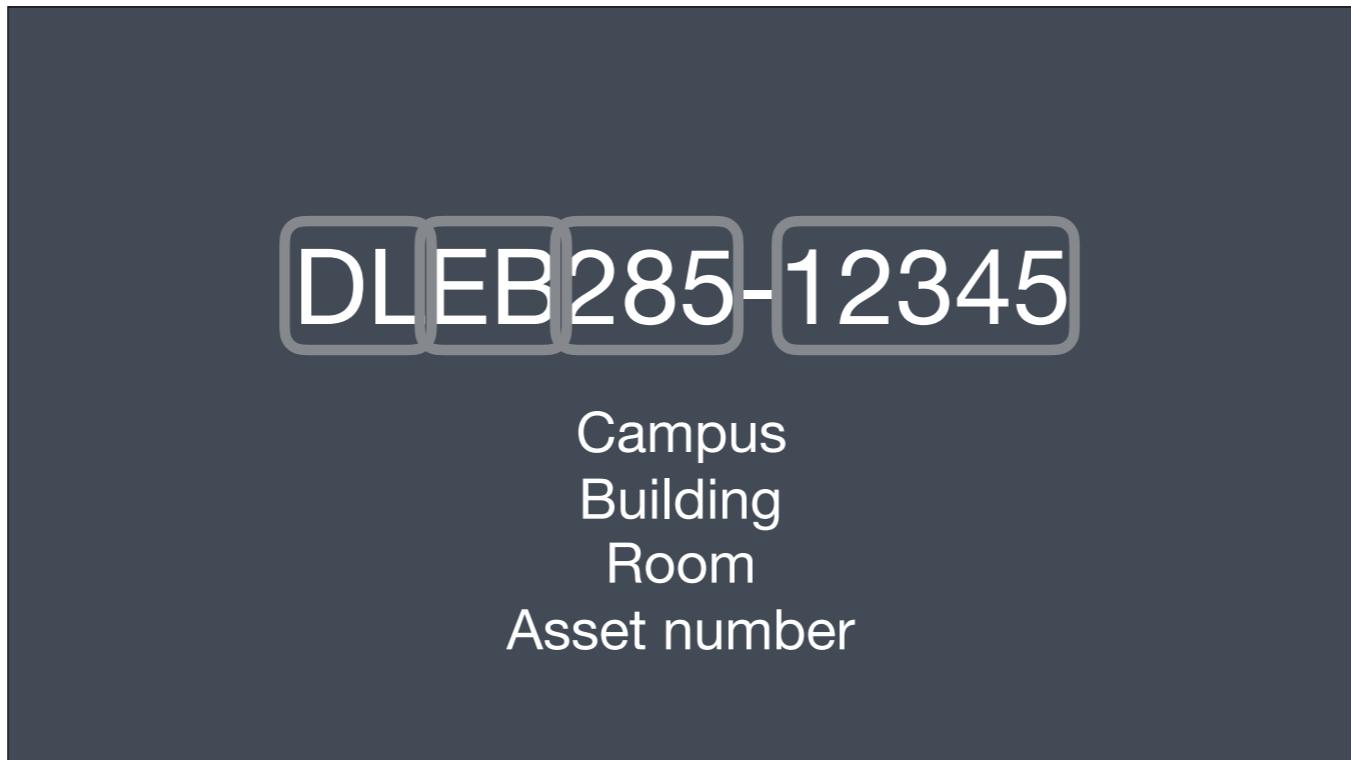
<C> have MDM profiles set most preferences

<C> Run a script to do everything else (like enable screen sharing, set the timezone and so on)

<C> and have policies to get all the software installed

The hostname is important to us...

And for it to work, we really need to set the hostname...



Because our hostnames tell us the <C> Campus (Docklands), <C> Building (East Building), <C>, Room number <C> and our own internal Asset number.

This means we can create smart groups, based on the computer name...

Computers > Smart Computer Groups >

All DL Student Macs

Computer Group	Criteria		
AND/OR	CRITERIA	OPERATOR	VALUE
	Mac User Role	is	Student
	Computer Name	like	DL

+ Add

Delete

...that have all the student facing Macs in a whole campus or building.

The screenshot shows a user interface for creating a smart computer group. The top navigation bar indicates the path: Computers > Smart Computer Groups > Lab DLEB240. The main section is titled "Criteria". A table lists one criterion: "Computer Name" with the operator "like" and the value "DLEB240". There are buttons for "Delete" and "+ Add" below the table.

AND/OR	CRITERIA	OPERATOR	VALUE
	Computer Name	like	DLEB240

+ Add

...or have all the Macs in a specific room or lab...

The screenshot shows a software interface for managing computer groups. At the top, there's a navigation bar with 'Computers' and 'Smart Computer Groups'. Below it, the title 'Deploy_DZED_Dragon_Frame' is displayed. There are two tabs: 'Computer Group' and 'Criteria', with 'Criteria' being the active tab. The main area is a table with columns: 'AND/OR', 'CRITERIA', 'OPERATOR', and 'VALUE'. The table contains three rows of criteria:

AND/OR	CRITERIA	OPERATOR	VALUE
	Application Title	is not	Dragonframe.app
and	(Computer Group	member of	Lab DLAVAG45
or	Computer Group	member of	Lab DLWBG05

At the bottom right of the table, there's a '+ Add' button.

When it comes to deploying an application, say Dragonframe, we could have a Smart Group that looks for Macs that are in the labs that require it but don't have it installed.

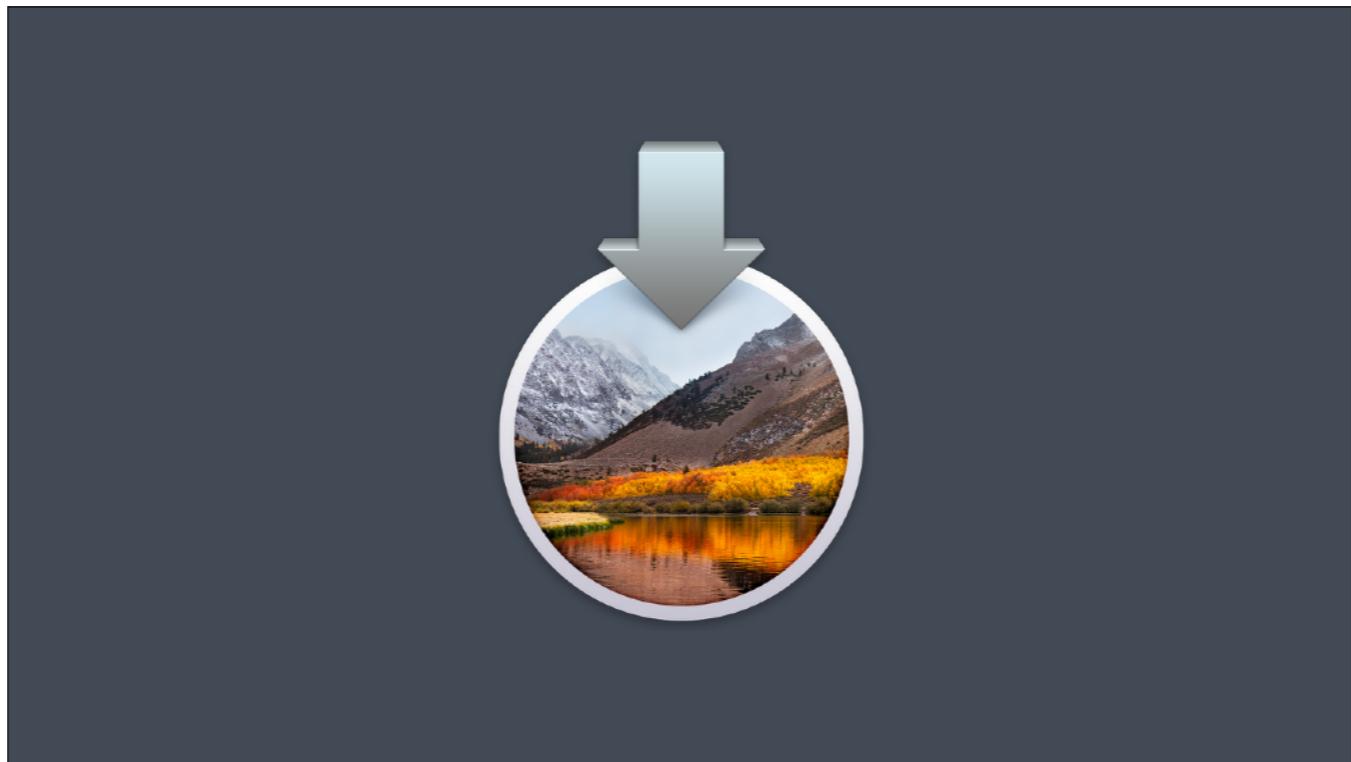
With this, we can scope a policy to this group that installs Dragonframe. If you want to deploy more applications, just create a similar smart group and policy for each one.

We also deal with version control for each application but that's a bit beyond the scope of this presentation.



Times were good...

Times were good!



Then High Sierra came along...

“

Apple doesn't recommend or support monolithic system imaging when upgrading or updating macOS.

- Apple, <https://support.apple.com/en-us/HT208020>

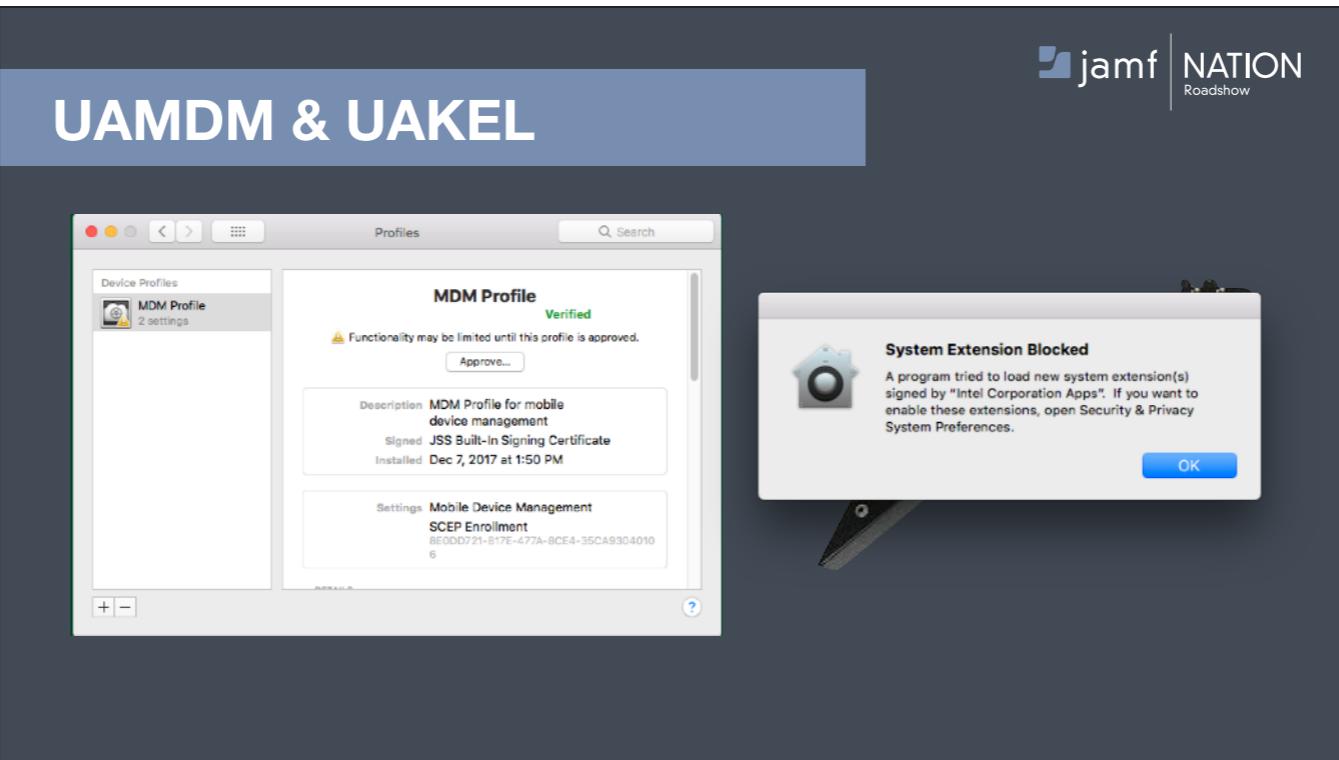
”

And Apple did this.

APFS and firmware



Because we have a new file system, that needs up to date firmware to boot from. The process of laying down an image doesn't install firmware.



If you image and your Macs are automatically enrolled into your MDM with your imaging tools, you'll have User Approved MDM to contend with. As well as User Approved Kernel Extension Loading.

<C> or Ukulele. You can't even click that Approve button remotely - you have to use a physical mouse on the actual Mac. Once you've approved MDM, you can push a profile to approve specific extensions, or the all the extensions from a specific vendor.

isimagingdead.com

?

So, is imaging dead?



It's looking that way.

Now what?

So, what can we do?

Stay on Sierra?

Could we stick with what we have?

Not really. When Apple release new Macs, you probably won't be able to downgrade them. And I don't think this is going to blow over...

In-place upgrade?

We could deploy 10.13 as an upgrade. You'd get the right firmware, you wouldn't have to deal with User Approved MDM and you wouldn't have to change your imaging workflows.

I don't particularly like this one either.

You end up with bits left over from the previous OS and some Apps that install things all over the place. What about new Macs?

Installation based workflow

+

DEP

So that leads us here. A clean installation of macOS, followed by enrolling it into Jamf through DEP. User Approved MDM is dealt with and we can take care of those kernel extensions with a profile.

Let's assume...



APNS



MDM

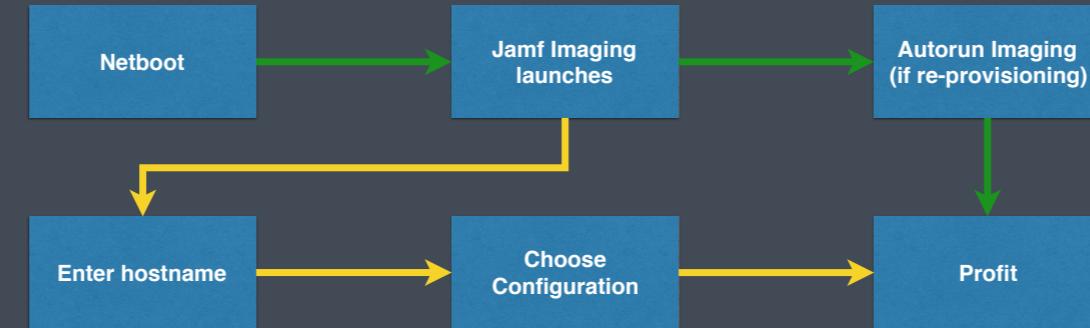


DEP/Apple School Manager

At this point I'm going to assume APNS is allowed over the network and that we're using MDM.

I'm also going to assume that we've just set up Apple School Manager (or Business Manager if you're not in education) and that our Macs are added to it.

Modular imaging workflow



Let's look at the old workflow again.

We need to complete most of the steps first...

Modular imaging workflow

Profit

...then the OS gets laid down at the end.

But with an install based workflow, we need to get the OS on there at the beginning...



Now, with High Sierra 10.13.4, Apple have given us a couple of ways to achieve that.

For the moment, you can still use NetBoot - create a NetInstall set with System Image Utility and set it up to automatically erase and install (this was broken with 10.13 but fixed in the update).

Or with Macs that have APFS volumes, get Apple's installer application onto them and trigger the `startosinstall` command with a new `--eraseinstall` flag.

PreStage Enrollment

General

DISPLAY NAME Display name for the PreStage enrollment

DEVICE ENROLLMENT PROGRAM INSTANCE
DEP instance to associate with the PreStage enrollment. Devices associated with the selected DEP instance can be assigned the PreStage enrollment

Automatically assign new devices
Automatically assign all new devices to this PreStage enrollment.

ENROLLMENT SITE Site that computers will be added to during enrollment

Use existing site membership, if applicable

Use existing location information, if applicable

SUPPORT PHONE NUMBER Support phone number for the organization

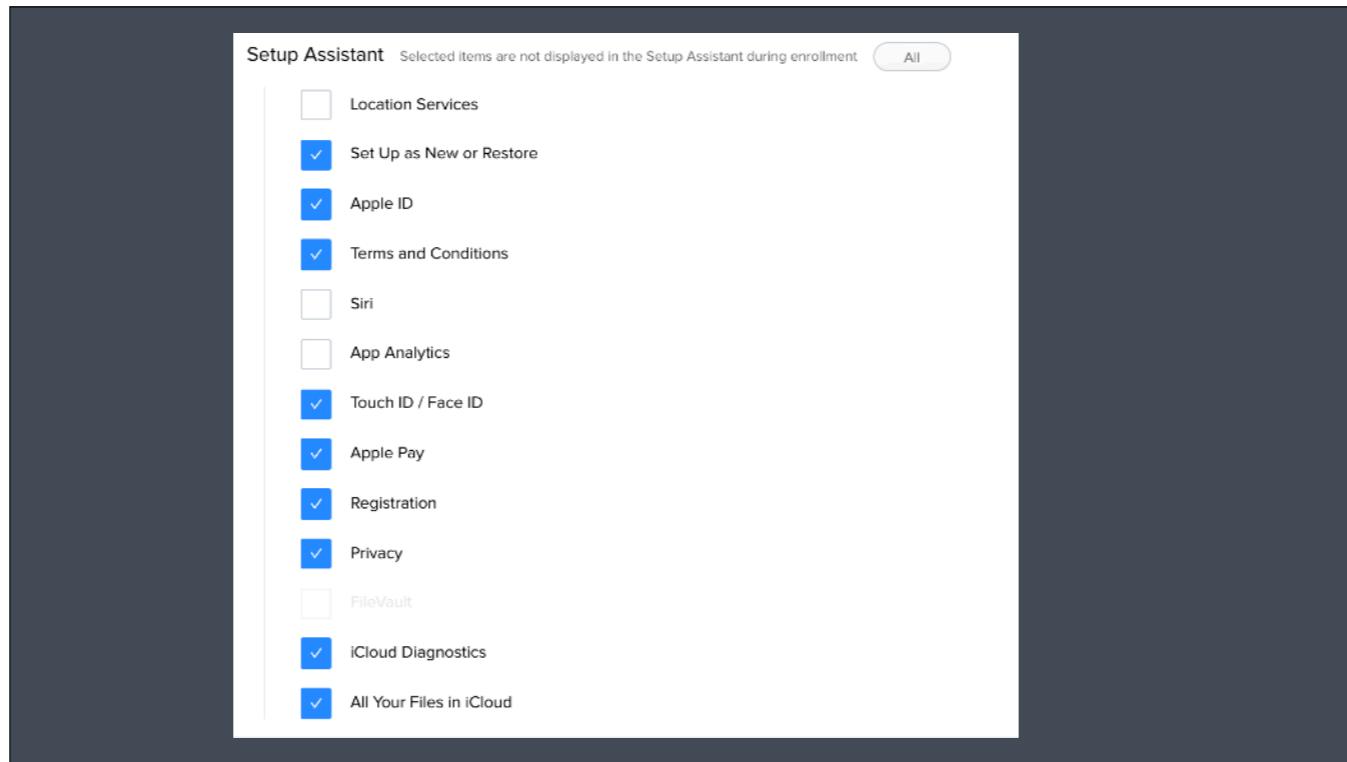
DEPARTMENT Department to associate with the PreStage enrollment

Require Authentication
Require the user to provide username and password on computers with macOS v10.10 or later

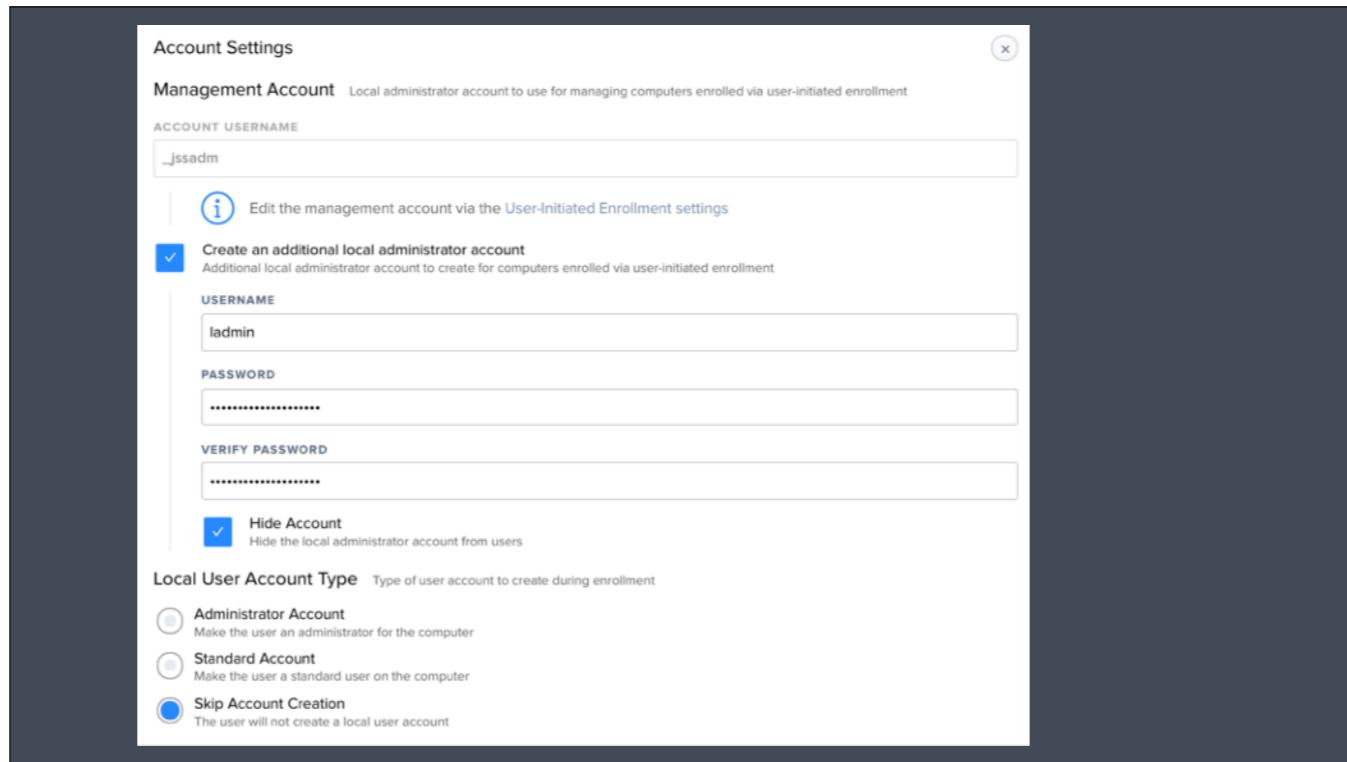
Make MDM Profile Mandatory
Require the user to apply the MDM profile

Allow MDM Profile Removal
Allow the user to remove the MDM profile

Next, You'd need to set up a PreStage Enrollment in Jamf



...this can skip the parts of the Setup Assistant you want to...



...and create a local administrator account.

Install/DEP workflow



Profit

In terms of our workflow, we get this far...

macOS gets installed and you can log in with the local admin account.

But

However...

The hostname is important to us...

We still want that hostname.

So is the role...

And we still want to set that role.

Because we still want to use our original smart group and policy workflows to determine which software gets installed.

So we need a way to collect that. Apple's Setup Assistant doesn't give us a way. But there is a tool that does...

Enter DEPNotify

Progress screen for DEP workflows

Optionally runs full screen

Driven by commands to a text file

Scrapes the jamf.log

User input - thanks to Frederico



DEPNotify, written by Joel Rennich who also wrote NoMAD.

<C> It's a small application that lets you know what's going on during a DEP based enrolment. And it's really easy to use.

<C> It can cover the whole screen and stop people messing about.

<C> and you can drive it to do different things during provisioning by echoing commands to a text file.

<C> It can also reads the Jamf log and update its status text when it sees policy executions or package installations.

<C> And just now, thanks to Frederico, DEPNotify can take input from the user and write that information to a plist, which we can read with other things.

Meanwhile, in #depnotify



fgd 01:34

Hey man, I already got a branch of DEPNotify that takes user input and uploads it to JSS and/or writes to a plist file. What info are you interested in capturing. I could customize the window for you.

 **neilmartin83**

Hey folks, dumb question - is anyone collecting user input and using DEPNotify? I pop up a couple of AppleScript dialogs to get the hostname and computer role (staff/student) - I'd like to have DEPNotify start full screen, then pop these dialogs up but they are opening behind DEPNotify - is there a way to get them to open on top of it?

Posted in #depnotify | Mar 2nd

Now I think this really highlights how great our community is.

You start with a silly question in the DEPNotify channel on the Mac Admins Slack...

One thing leads to another...

Meanwhile, in #depnotify



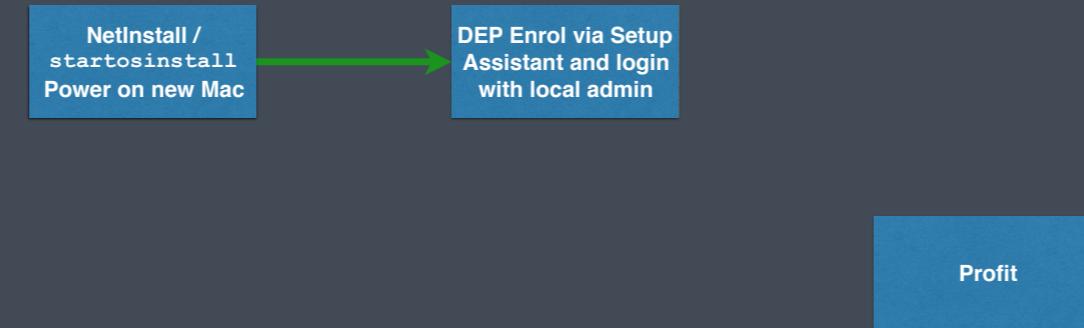
fgd 00:02

So I'm working on a version of DEPNotify that will accept user input for 2 text fields and two pop up menus, all customizable by defaults keys. You'd be able to call the user input dialog at any time with a button trigger. Labels, placeholders, text field character compliance, popup menu content, window title, path to save a plist file, etc. will come from keys in a prefs file. One would also be able to customize which input fields to show. The same would work for the agreement window. This version of DEPNotify will be easy to integrate with NoNotify to be run on top of the login window.



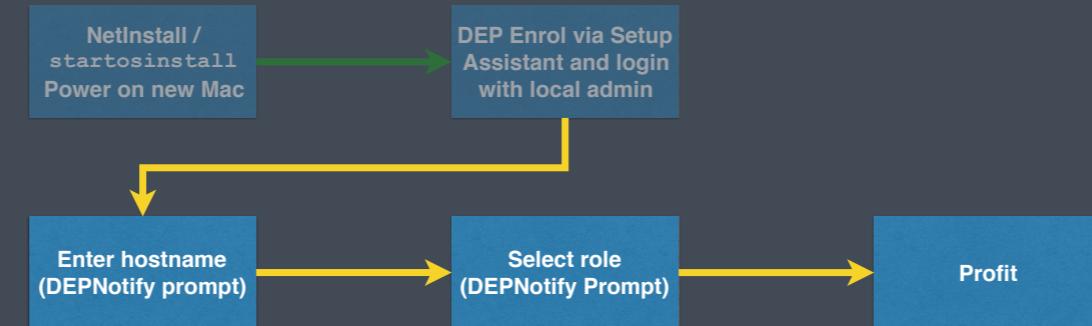
...and after lots of testing, more discussion, and even more testing great things start to happen.

Install/DEP workflow



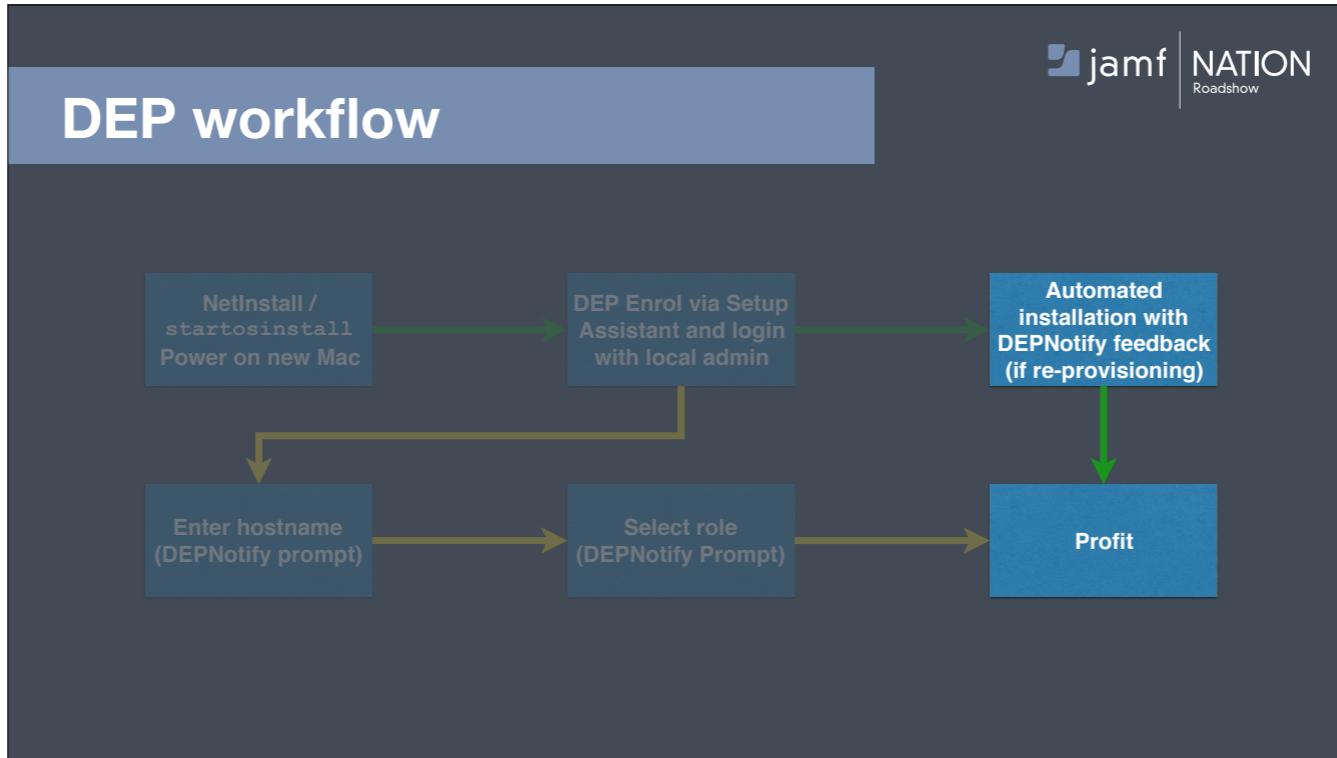
So let's add the missing pieces to our workflow.

DEP workflow



We need to set the hostname and role.

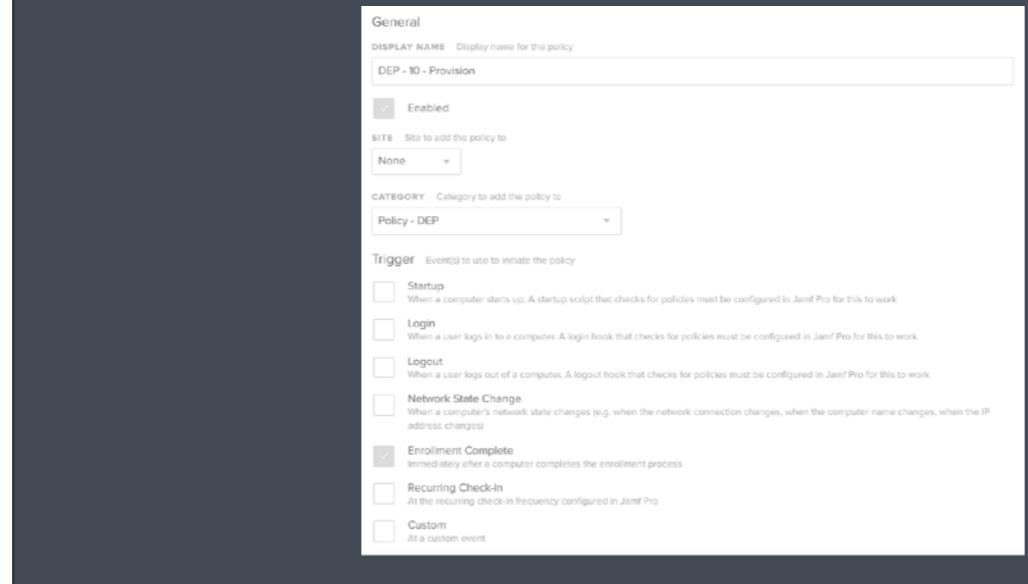
DEP workflow



But if we're re-provisioning a Mac that's already in Jamf, we want to skip collecting that data and reuse what we have.

Like we did with autorun imaging.

Policy - Enrolment Complete



General

DISPLAY NAME: DEP - 10 - Provision

Enabled

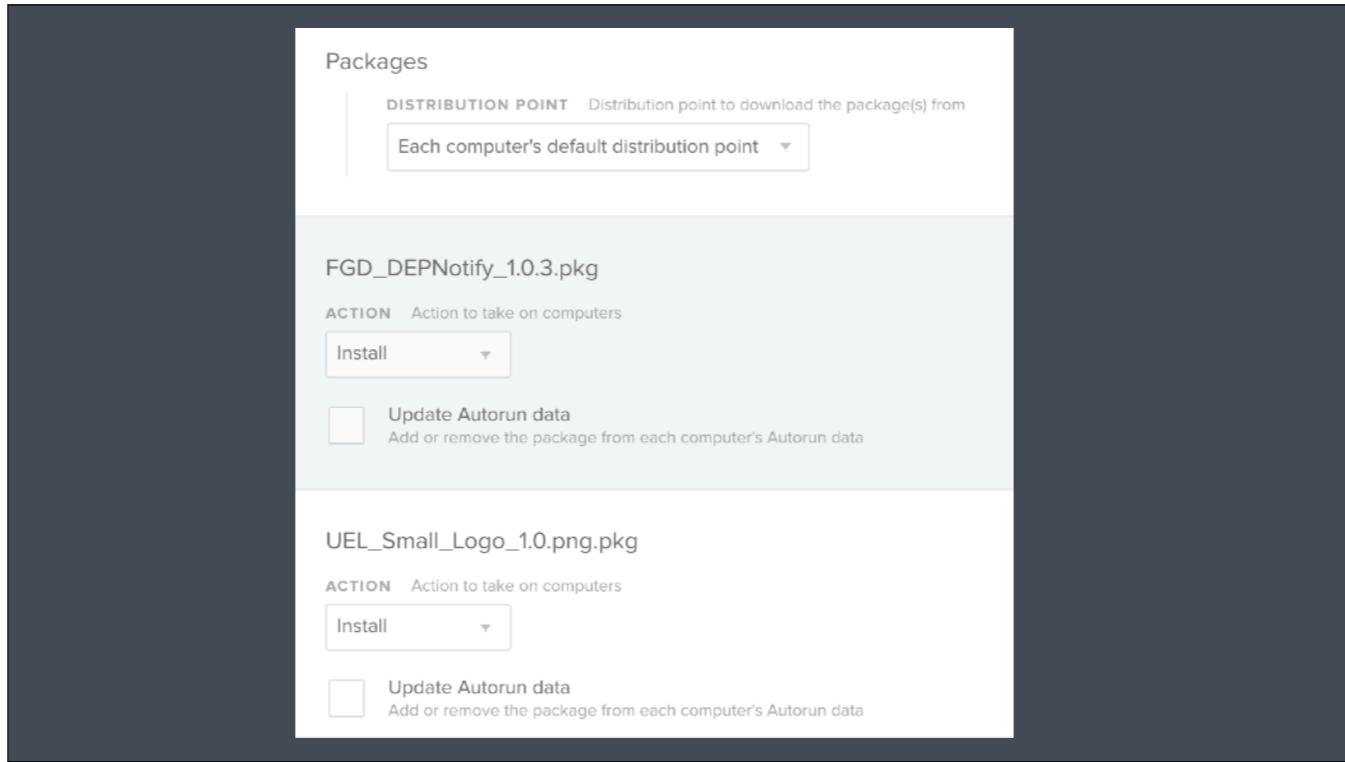
SITE: None

CATEGORY: Policy - DEP

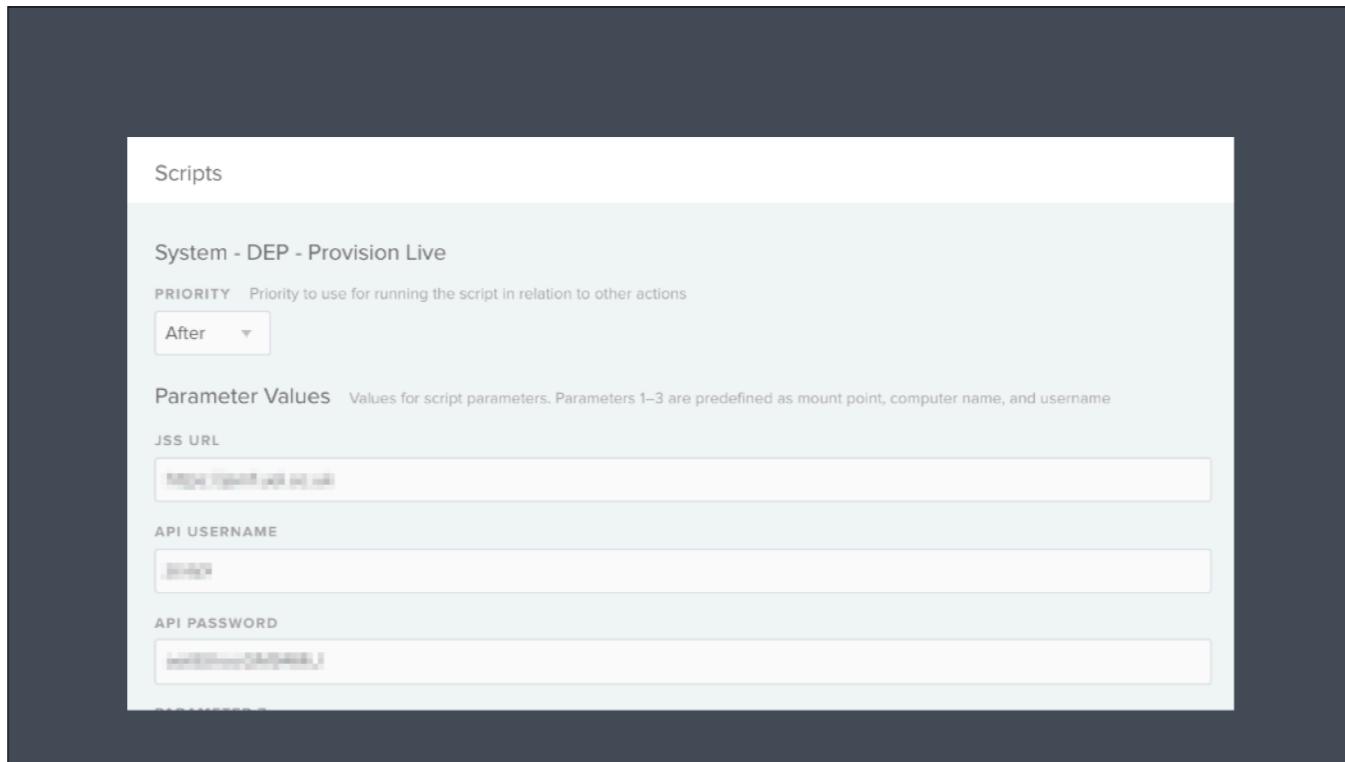
Trigger

- Startup
- Login
- Logout
- Network State Change
- Enrollment Complete
- Recurring Check-in
- Custom

We need a policy that runs on the Enrolment Complete trigger.



This will install DEPNotify and our own branding icon for it.



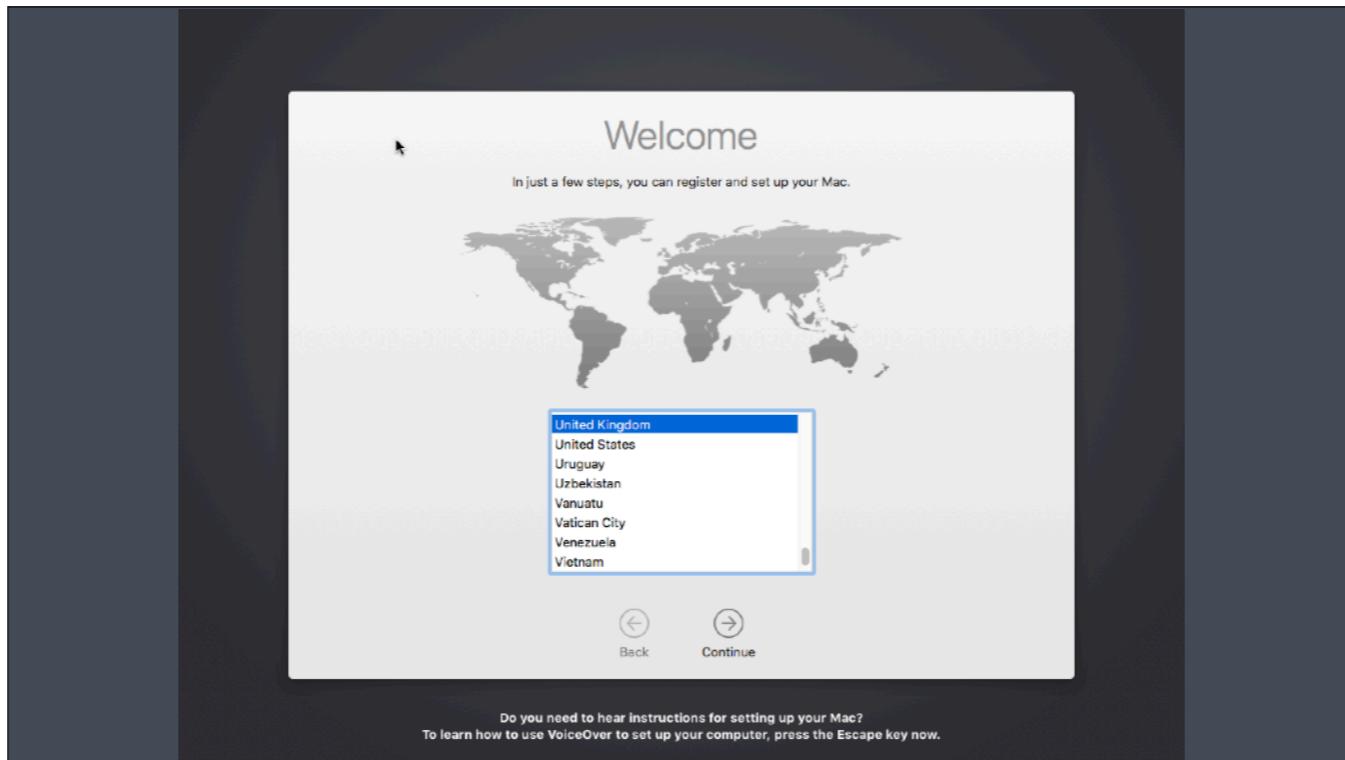
And run a script. As you can see here, we're using a few parameters.

In this case, our account credentials for API access.

This means you don't have to hard code your API account username and password in the actual script.

What does it look like?

Lets put it all together.

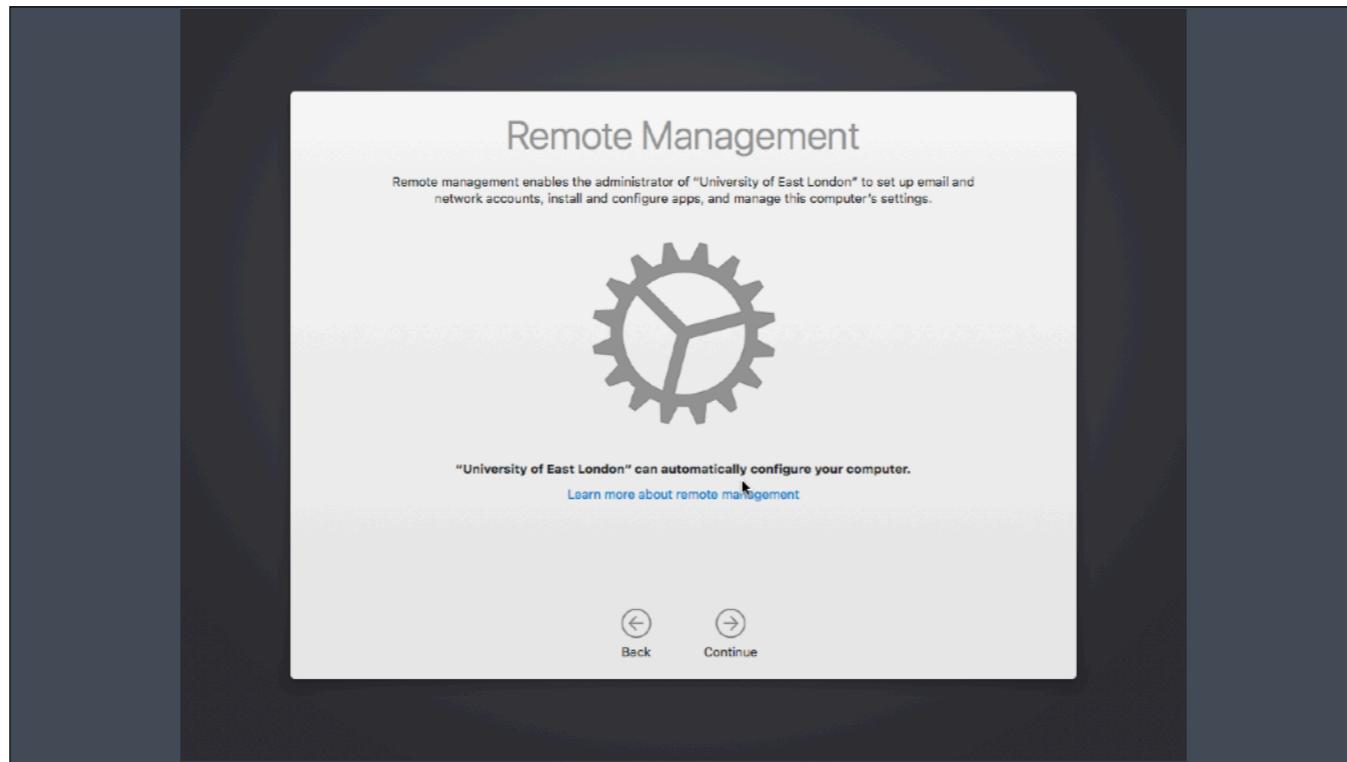


You've installed macOS and we're at the Setup Assistant

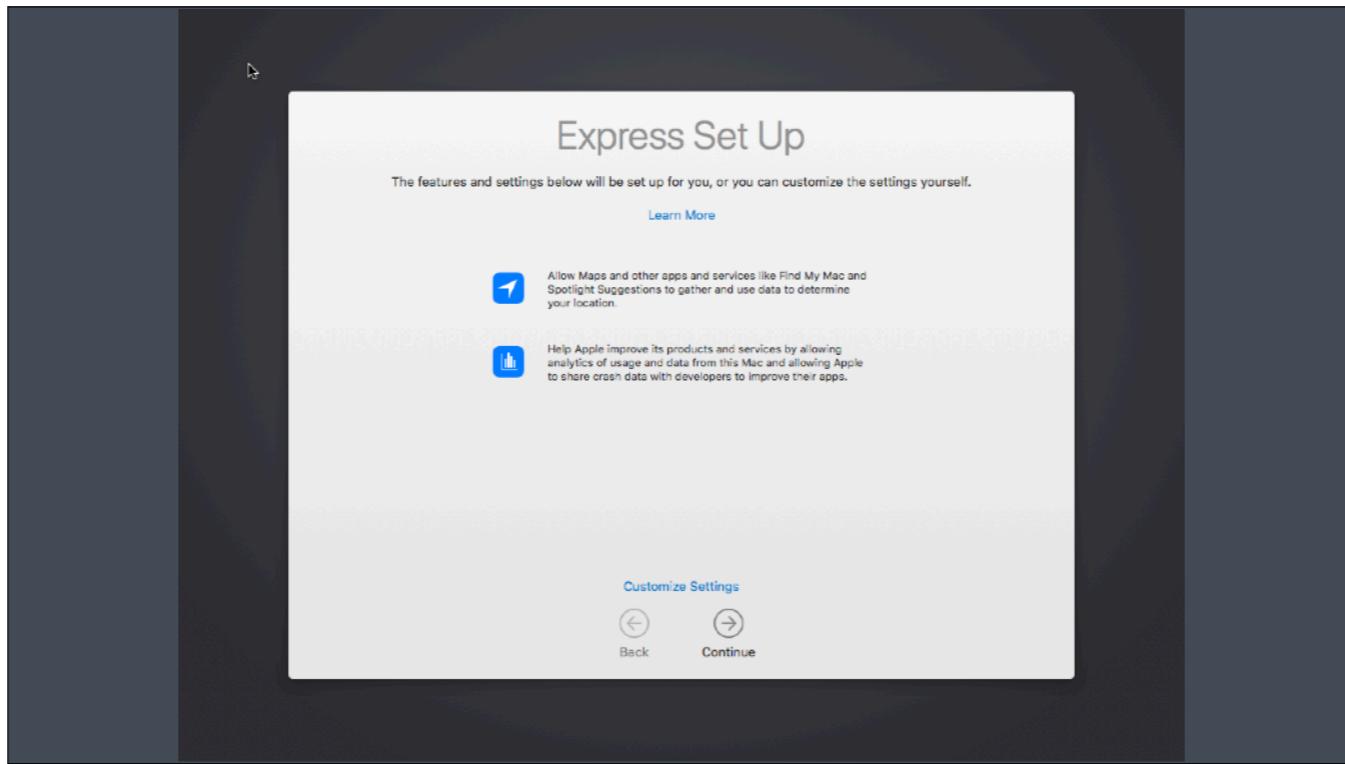
Choose your country



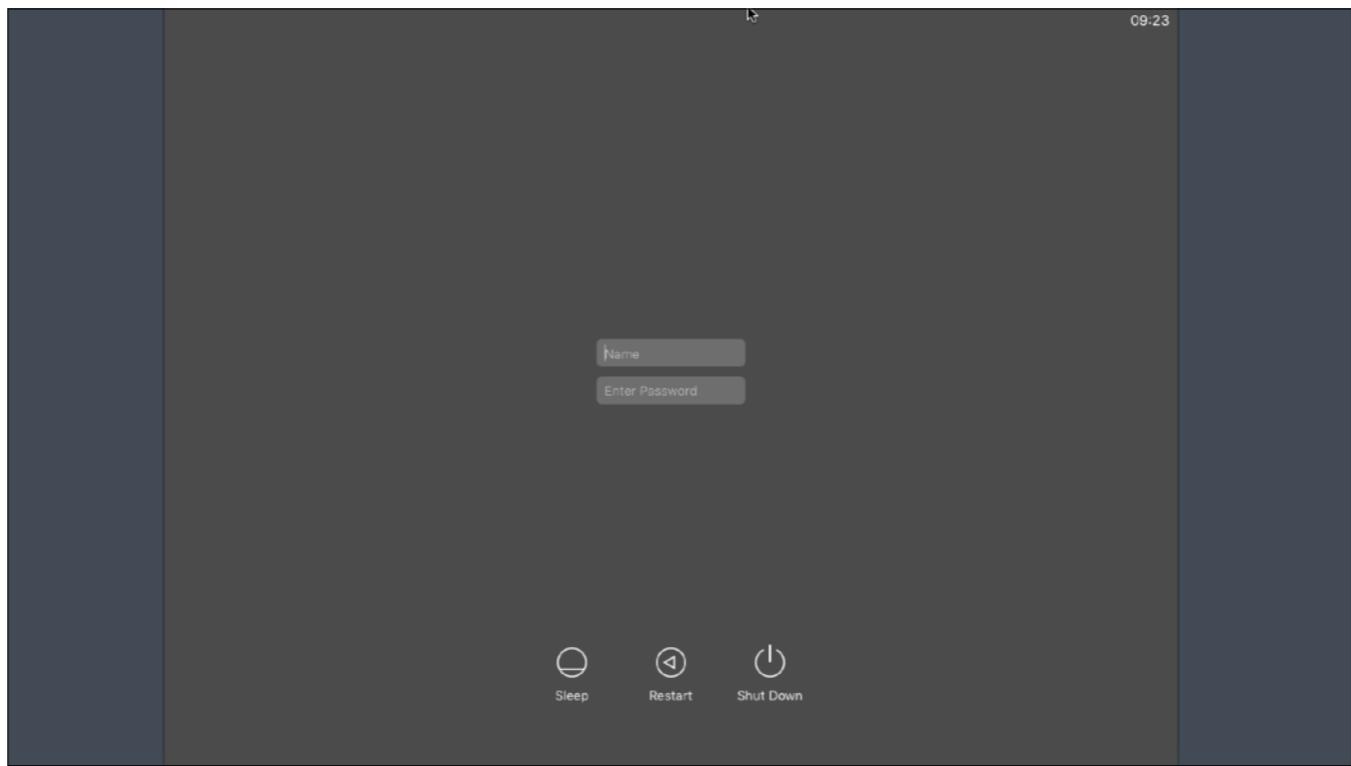
Your keyboard layout



Hit the DEP screen



Proceed through the rest of the Setup Assistant



Login as local admin



And off we go...

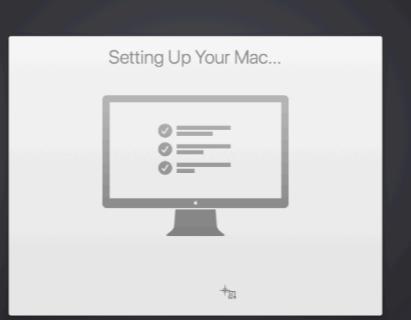
```
1 #!/bin/bash
2
3 # $4 = JSS URL
4 # $5 = JSS account username for API access
5 # $6 = JSS account password for API access
6
7 # Set basic variables
8 osversion=$(sw_vers -productVersion)
9 serial=$(ioreg -rd1 -c IOPlatformExpertDevice | awk -F'"' '/IOPlatformSerialNumber/{print $4}')
10
11 # Let's not go to sleep
12 caffeinate -d -l -m -s -u &
13 caffeinatepid=$!
14
15 # Disable Software Updates during imaging
16 softwareupdate --schedule off
17
18 dockStatus=$(pgrep -x Dock)
19 while [[ "$dockStatus" == "" ]]; do
20     sleep 5
21     dockStatus=$(pgrep -x Dock)
22 done
23
24 # Get the currently logged in user
25 loggedInUser=$(python -c 'from SystemConfiguration import SCDynamicStoreCopyConsoleUser; print(SCDynamicStoreCopyConsoleUser(None, None))')
26
27 # Check for existing Hostname exten-
# otherwise, automation baby!
28
29 jssHostName=$(curl "$4":8443/JSSRe
| xpath '//extension_attribute[nam
30 jssUserRole=$(curl "$4":8443/JSSRe
| xpath '//extension_attribute[nam
```



```
eUser; import sys; username =
ime,""])[username in [u"loginwindow",
l ask for the name and role,
extension_attributes --user "$5":"$6"
:$2'})")
extension_attributes --user "$5":"$6"
print $2'})")
```

Our Enrolment Complete policy fires, installing DEPNotify and running this script

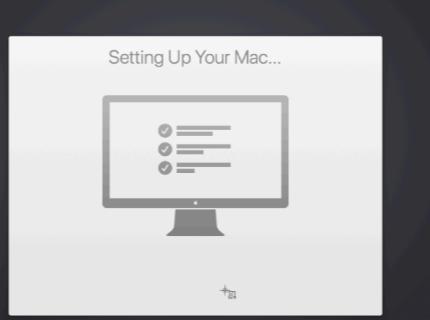
```
1 #!/bin/bash
2
3 # $4 = JSS URL
4 # $5 = JSS account username for API access
5 # $6 = JSS account password for API access
6
7 # Set basic variables
8 osversion=$(sw_vers -productVersion)
9 serial=$(ioreg -rd1 -c IOPlatformExpertDevice | awk -F"'+' '/IOPlatformSerialNumber/{print $4}')
10
11 # Let's not go to sleep
12 caffeinate -d -l -m -s -u &
13 caffeinatepid=$!
14
15 # Disable Software Updates during imaging
16 softwareupdate --schedule off
17
18 dockStatus=$(pgrep -x Dock)
19 while [[ "$dockStatus" == "" ]]; do
20     sleep 5
21     dockStatus=$(pgrep -x Dock)
22 done
23
24 # Get the currently logged in user
25 loggedInUser=$(python -c 'from SystemConfiguration import SCDynamicStoreCopyConsoleUser; print(SCDynamicStoreCopyConsoleUser(None, None))')
26
27 # Check for existing Hostname exten-
# otherwise, automation baby!
28
29 jssHostName=$(curl "$4":8443/JSSRe-
| xpath '//extension_attribute[nam-
30 jssUserRole=$(curl "$4":8443/JSSRe-
| xpath '//extension_attribute[nam-
```



```
Setting Up Your Mac...
eUser; import sys; username =
ime,""])[username in [u"loginwindow",
l ask for the name and role,
xtension_attributes --user "$5":"$6"
:$2'})'
xtension_attributes --user "$5":"$6"
print $2'})'
```

First we grab the OS version and serial number and store them as a couple of variables

```
1 #!/bin/bash
2
3 # $4 = JSS URL
4 # $5 = JSS account username for API access
5 # $6 = JSS account password for API access
6
7 # Set basic variables
8 osversion=$(sw_vers -productVersion)
9 serial=$(ioreg -rd1 -c IOPlatformExpertDevice | awk -F'"' '/IOPlatformSerialNumber/{print $4}')
10
11# Let's not go to sleep
12caffeinate -d -l -m -s -u &
13caffeinatepid=$!
14
15# Disable Software Updates during imaging
16softwareupdate --schedule off
17
18dockStatus=$(pgrep -x Dock)
19while [[ "$dockStatus" == "" ]]; do
20    sleep 5
21    dockStatus=$(pgrep -x Dock)
22done
23
24# Get the currently logged in user
25loggedInUser=$(python -c 'from SystemConfiguration import SCDynamicStoreCopyConsoleUser; print(SCDynamicStoreCopyConsoleUser(None, None))')
26
27# Check for existing Hostname exten-
28# otherwise, automation baby!
29
30jssHostName=$(curl "$4":8443/JSSRe
| xpath '//extension_attribute[nam
30jssUserRole=$(curl "$4":8443/JSSRe
| xpath '//extension_attribute[nam
```



The screenshot shows a Mac desktop with a window titled "Setting Up Your Mac...". Inside the window, there is a graphic of a computer monitor displaying a simple interface with three circular icons.

```
eUser; import sys; username =
ime,""])[username in [u"loginwindow",
l ask for the name and role,
extension_attributes --user "$5":"$6"
:$2'})")
extension_attributes --user "$5":"$6"
print $2'})"
```

We don't want the Mac to sleep, so we deal with that.

```
1 #!/bin/bash
2
3 # $4 = JSS URL
4 # $5 = JSS account username for API access
5 # $6 = JSS account password for API access
6
7 # Set basic variables
8 osversion=$(sw_vers -productVersion)
9 serial=$(ioreg -rd1 -c IOPlatformExpertDevice | awk -F'"' '/IOPlatformSerialNumber/{print $4}')
10
11 # Let's not go to sleep
12 caffeinate -d -l -m -s -u &
13 caffeinatepid=$!
14
15 # Disable Software Updates during imaging
16 softwareupdate --schedule off
17
18 dockStatus=$(pgrep -x Dock)
19 while [[ "$dockStatus" == "" ]]; do
20     sleep 5
21     dockStatus=$(pgrep -x Dock)
22 done
23
24 # Get the currently logged in user
25 loggedInUser=$(python -c 'from SystemConfiguration import SCDynamicStoreCopyConsoleUser; print(SCDynamicStoreCopyConsoleUser(None, None))')
26
27 # Check for existing Hostname exten-
# otherwise, automation baby!
28
29 jssHostName=$(curl "$4":8443/JSSRe
| xpath '//extension_attribute[nam
30 jssUserRole=$(curl "$4":8443/JSSRe
| xpath '//extension_attribute[nam
```



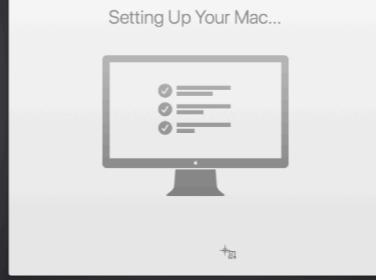
```
eUser; import sys; username =
ime,""])[username in [u"loginwindow",
l ask for the name and role,
extension_attributes --user "$5":"$6"
$2'})")
extension_attributes --user "$5":"$6"
print $2'})")
```

We nip any software update checks in the bud.

```

17 dockStatus=$(pgrep -x Dock)
18 while [[ "$dockStatus" == "" ]]; do
19     sleep 5
20     dockStatus=$(pgrep -x Dock)
21 done
23
24 # Get the currently logged in user's username
25 loggedInUser=$(python -c 'from SystemConfiguration import SCDynamicStoreCopyConsoleUser; import sys; username = (SCDynamicStoreCopyConsoleUser(None, None, None) or [None])[0]; username = [username,""]るusername in [u"loginwindow", None, u""]); sys.stdout.write(username + "\n");')
26
27 # Check for existing Hostname extension attribute in JSS - if it's not there, we'll ask for the name and role,
# otherwise, automation baby!
28
29 jssHostName=$(curl "$4":8443/JSSResource/computers/serialnumber/"$serial"/subset/extension_attributes --user "$5":"$6"
| xpath '//extension_attribute[name="Hostname"]' | awk -F'<value>|</value>' '{print $2}')
30 jssUserRole=$(curl "$4":8443/JSSResource/computers/serialnumber/"$serial"/subset/extension_attributes --user "$5":"$6"
| xpath '//extension_attribute[name="Mac User Role"]' | awk -F'<value>|</value>' '{print $2}')
31
32 if [[ "$jssHostName" == "" ]] || [
33     sudo -u "$loggedInUser" default
34     sudo -u "$loggedInUser" default
35     sudo -u "$loggedInUser" default
36     sudo -u "$loggedInUser" default
37     sudo -u "$loggedInUser" default
38     sudo -u "$loggedInUser" default
39     sudo -u "$loggedInUser" default
40     echo "Command: ContinueButtonR"
41     echo "Command: Image: /Library/PrivilegedHelperTools/depnotify"
42     echo 'Command: MainTitle: Hi there'
43     echo "Command: MainText: It's time to log in. Please make sure it is plugged in. Contact the UEL IT Service Desk if you need any assistance. Please set the computer name and role in Jamf Pro before continuing. If you need any assistance, please contact support@uel.ac.uk" >>
44     echo "Status: Please set the computer name and role in Jamf Pro before continuing. If you need any assistance, please contact support@uel.ac.uk"

```



```

ar/tmp/
"Let's get started..."
Label OK
Placeholder "DLEB123-12345"
bel "Computer Name"
array Student "Staff" "Staff Loan"
bel "Computer Role"
mp/depnotify.log

ettings it needs. Before we continue,
If you need any assistance, please
contact support@uel.ac.uk" >>
in/depnotify.log

```

And we wait for login to complete.

This is important - because of the way Jamf does enrolment with DEP, the script might have started before we logged in. DEPNotify needs to run after login.

One way to detect login is to wait for the Dock process with a while loop.

```
17
18 dockStatus=$(pgrep -x Dock)
19 while [[ "$dockStatus" == "" ]]; do
20     sleep 5
21     dockStatus=$(pgrep -x Dock)
22 done
23
24 # Get the currently logged in user's username
25 loggedInUser=$(python -c 'from SystemConfiguration import SCDynamicStoreCopyConsoleUser; import sys; username = (SCDynamicStoreCopyConsoleUser(None, None, None) or [None])[0]; username = [username,""] if username in [u"loginwindow", None, u""]]; sys.stdout.write(username + "\n");')
26
27 # Check for existing Hostname extension attribute in JSS - if it's not there, we'll ask for the name and role,
# otherwise, automation baby!
28
29 jssHostName=$(curl "$4":8443/JSSResource/computers/serialnumber/"$serial"/subset/extension_attributes --user "$5":"$6"
# | xpath '//extension_attribute[name="Hostname"]' | awk -F'<value>|</value>' '{print $2}')
30 jssUserRole=$(curl "$4":8443/JSSResource/computers/serialnumber/"$serial"/subset/extension_attributes --user "$5":"$6"
# | xpath '//extension_attribute[name="Mac User Role"]' | awk -F'<value>|</value>' '{print $2}')
31
32 if [[ "$jssHostName" == "" ]] || [
33     sudo -u "$loggedInUser" defaultLabel "DLEB123-12345"
34     sudo -u "$loggedInUser" defaultPlaceholder "Computer Name"
35     sudo -u "$loggedInUser" defaultArray "Student" "Staff" "Staff Loan"
36     sudo -u "$loggedInUser" defaultLabel "Computer Role"
37     sudo -u "$loggedInUser" defaultCommand "/Library/Application Support/depnotify/depnotify -f /var/tmp/depnotify.log"
38     sudo -u "$loggedInUser" defaultImage "/Library/Application Support/depnotify/Images/DepNotifyImage.png"
39     sudo -u "$loggedInUser" defaultMainText "Hi there! This is a test message from depnotify."
40     echo "Command: ContinueButtonRole"
41     echo "Command: Image: /Library/Application Support/depnotify/Images/DepNotifyImage.png"
42     echo 'Command: MainTitle: Hi there'
43     echo "Command: MainText: It's time to log in now. Please make sure your computer is plugged in and has the correct settings it needs. Before we continue, please make sure you have the latest version of depnotify installed. If you need any assistance, please contact the UEL IT Service Desk at icedesk@uel.ac.uk" >>
44     echo "Status: Please set the correct computer name and role in the depnotify configuration file." >> /var/tmp/depnotify.log
echo "Status: Please set the correct computer name and role in the depnotify configuration file." >> /var/tmp/depnotify.log
```

And we're logged in. Let's get the logged in user's username.

```
28
29 jssHostName=$(curl "$4":8443/JSSResource/computers/serialnumber/"$serial"/subset/extension_attributes --user "$5":"$6"
| xpath '//extension_attribute[name="Hostname"]' | awk -F'<value>|</value>' '{print $2}')
30 jssUserRole=$(curl "$4":8443/JSSResource/computers/serialnumber/"$serial"/subset/extension_attributes --user "$5":"$6"
| xpath '//extension_attribute[name="Mac User Role"]' | awk -F'<value>|</value>' '{print $2}')
31
32 if [[ "$jssHostName" == "" ]] || [[ "$jssUserRole" == "" ]]; then
33   sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify PathToPlistFile /var/tmp/
34   sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify RegisterMainTitle "Let's get started..."
35   sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify RegistrationButtonLabel OK
36   sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify UITextFieldUpperPlaceholder "DLEB123-12345"
37   sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify UITextFieldUpperLabel "Computer Name"
38   sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify UIPopUpMenuUpper -array Student "Staff" "Staff Loan"
39   sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify UIPopUpMenuUpperLabel "Computer Role"
40   echo "Command: ContinueButtonRegister: Continue" >> /var/tmp/deponotify.log
41   echo "Command: Image: "/Library/Application Support/UEL/ux/UEL.png"" >> /var/tmp/deponotify.log
42   echo 'Command: MainTitle: Hi there!' >> /var/tmp/deponotify.log
43   echo "Command: MainText: It's time to set up this Mac with the software and settings it needs. Before we continue,
please make sure it is plugged into a wired network connection on campus. \n\n If you need any assistance, please
contact the UEL IT Service Des /var/tmp/deponotify.log
44   echo "Status: Please set the c /var/tmp/deponotify.log
45   sudo -u "$loggedInUser" /Appl /var/tmp/deponotify.log
46   sleep 1
47
48 # Wait for the user data to be
49 while [ ! -f /var/tmp/DEPNotify
50   echo "Status: Please set t /var/tmp/deponotify.log
51   sleep 5
52 done
53
54 # Let's read the user data into
55 computerName=$(/usr/libexec/pl /var/tmp/deponotify.log
56 computerRole=$(/usr/libexec/pl /var/tmp/deponotify.log
57
58 # Update Hostname and Computer
```



And let's do a 2 API calls to read back our hostname and role Extension Attributes, which will only be there if the Mac already has a computer record in Jamf and we're re-provisioning it.

```
31
32 if [[ "$jssHostName" == "" ]] || [[ "$jssUserRole" == "" ]]; then
33   sudo -u $loggedInUser defaults write menu.nomad.DEPNotify PathToPlistFile /var/tmp/
34   sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify RegisterMainTitle "Let's get started..."
35   sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify RegistrationButtonLabel OK
36   sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify UITextFieldUpperPlaceholder "DLEB123-12345"
37   sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify UITextFieldUpperLabel "Computer Name"
38   sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify UIPopUpMenuUpper -array Student "Staff" "Staff Loan"
39   sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify UIPopUpMenuUpperLabel "Computer Role"
40   echo "Command: ContinueButtonRegister: Continue" >> /var/tmp/depnotify.log
41   echo "Command: Image: '/Library/Application Support/UEL/ux/UEL.png'" >> /var/tmp/depnotify.log
42   echo 'Command: MainTitle: Hi there!' >> /var/tmp/depnotify.log
43   echo "Command: MainText: It's time to set up this Mac with the software and settings it needs. Before we continue,
please make sure it is plugged into a wired network connection on campus. \n \n If you need any assistance, please
contact the UEL IT Service Desk. \n \n Telephone: 020 8223 2558 \n Email: servicedesk@uel.ac.uk" >>
/var/tmp/depnotify.log
44   echo "Status: Please set the computer name and role to continue..." >> /var/tmp/depnotify.log
45   sudo -u "$loggedInUser" /Applications/Utilities/DEPNotify.app/Contents/MacOS/DEPNotify -jamf -fullScreen &
46   sleep 1
47
48 # Wait for the user data to be entered
49 while [ ! -f /var/tmp/DEPNotify ]
50   echo "Status: Please set the computer name and role to continue..." >> /var/tmp/depnotify.log
51   sleep 5
52 done
53
54 # Let's read the user data into variables
55 computerName=$(/usr/libexec/pluto/extension_attributes.sh get "Computer Name")
56 computerRole=$(/usr/libexec/pluto/extension_attributes.sh get "Computer Role")
57
58 # Update Hostname and Computer Role
59 # Create xml
60 cat << EOF > /var/tmp/name.xml
61 <computer>
62   <extension_attributes>
63     <extension_attributes>
```



If either of those Extension Attributes are empty, we're going to set up DEPNotify to prompt for user input.

```
31
32 if [[ "$isHostName" == "" ]] || [[ "$isUserRole" == "" ]]; then
33     sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify PathToPlistFile /var/tmp/
34     sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify RegisterMainTitle "Let's get started..."
35     sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify RegistrationButtonLabel OK
36     sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify UITextFieldUpperPlaceholder "DLEB123-12345"
37     sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify UITextFieldUpperLabel "Computer Name"
38     sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify UIPopUpMenuUpper -array Student "Staff" "Staff Loan"
39     sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify UIPopUpMenuItemUpperLabel "Computer Role"
40     echo "Command: ContinueButtonRegister: Continue" >> /var/tmp/deponotify.log
41     echo "Command: Image: "/Library/Application Support/UEL/ux/UEL.png" >> /var/tmp/deponotify.log
42     echo 'Command: MainTitle: Hi there!' >> /var/tmp/deponotify.log
43     echo "Command: MainText: It's time to set up this Mac with the software and settings it needs. Before we continue,
44     please make sure it is plugged into a wired network connection on campus. \n \n If you need any assistance, please
45     contact the UEL IT Service Desk. \n \n Telephone: 020 8223 2558 \n Email: servicedesk@uel.ac.uk" >>
46     /var/tmp/deponotify.log
47     echo "Status: Please set the computer name and role to continue..." >> /var/tmp/deponotify.log
48     sudo -u "$loggedInUser" /Applications/Utilities/DEPNotify.app/Contents/MacOS/DEPNotify -jamf -fullScreen &
49     sleep 1
50
51     # Wait for the user data to be entered
52     while [ ! -f /var/tmp/DEPNotify.plist ]; do
53         echo "Status: Please set the computer name and role to continue..." >> /var/tmp/deponotify.log
54         sleep 5
55     done
56
57     # Let's read the user data into variables
58     computerName=$(/usr/libexec/plist/libxml/parsePlist /var/tmp/DEPNotify.plist | grep -A 1 "computerName")
59     computerRole=$(/usr/libexec/plist/libxml/parsePlist /var/tmp/DEPNotify.plist | grep -A 1 "computerRole")
60
61     # Update Hostname and Computer Role
62     # Create xml
63     cat << EOF > /var/tmp/name.xml
64     <computer>
65         <extension_attributes>
66             <extension_attributes>
```



We write some preferences to it - we're defining where it'll put the resultant plist file, the title of the dialog along with a text field for the hostname and pop up menu for the role.

```
31
32 if [[ "$jssHostName" == "" ]] || [[ "$jssUserRole" == "" ]]; then
33   sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify PathToPlistFile /var/tmp/
34   sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify RegisterMainTitle "Let's get started..."
35   sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify RegistrationButtonLabel OK
36   sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify UITextFieldUpperPlaceholder "DLEB123-12345"
37   sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify UITextFieldUpperLabel "Computer Name"
38   sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify UIPopUpMenuUpper -array Student "Staff" "Staff Loan"
39   sudo -u "$loggedInUser" defaults write menu.nomad.DEPNotify UIPopUpMenuUpperLabel "Computer Role"
40
41 echo "Command: ContinueButtonRegister: Continue" >> /var/tmp/deponotify.log
42 echo "Command: Image: "/Library/Application Support/UEL/ux/UEL.png" >> /var/tmp/deponotify.log
43 echo 'Command: MainTitle: Hi there!' >> /var/tmp/deponotify.log
44 echo "Command: MainText: It's time to set up this Mac with the software and settings it needs. Before we continue,
please make sure it is plugged into a wired network connection on campus. \n \n If you need any assistance, please
contact the UEL IT Service Desk. \n \n Telephone: 020 8223 2558 \n Email: servicedesk@uel.ac.uk" >>
/var/tmp/deponotify.log
45 echo "Status: Please set the computer name and role to continue..." >> /var/tmp/deponotify.log
46 sudo -u "$loggedinUser" /Applications/Utilities/DEPNotify.app/Contents/MacOS/DEPNotify -jamf -fullScreen &
47 sleep 1
48
49 # Wait for the user data to be entered
50 while [ ! -f /var/tmp/DEPNotify ]
51   echo "Status: Please set the computer name and role to continue..." >> /var/tmp/deponotify.log
52   sleep 5
53 done
54
55 # Let's read the user data into variables
56 computerName=$(/usr/libexec/pluto/GetComputerName)
57 computerRole=$(/usr/libexec/pluto/GetComputerRole)
58
59 # Update Hostname and Computer Role
60 # Create xml
61 cat << EOF > /var/tmp/name.xml
62 <computer>
63   <extension_attributes>
64     <extension_attributes>
```

Next, we're echoing commands so when DEPNotify launches, it will give us a button, use our icon and have the title, main text and status text we want.

```
44 echo "Status: Please set the computer name and role to continue..." >> /var/tmp/depnotify.log
45 sudo -u "$loggedInUser" /Applications/Utilities/DEPNotify.app/Contents/MacOS/DEPNotify -jamf -fullScreen &
46 sleep 1
47
48 # Wait for the user data to be submitted...
49 while [ ! -f /var/tmp/DEPNotify.plist ]; do
50     echo "Status: Please set the computer name and role to continue..." >> /var/tmp/depnotify.log
51     sleep 5
52 done
53
54 # Let's read the user data into some variables...
55 computerName=$(/usr/libexec/plistbuddy /var/tmp/DEPNotify.plist -c "print 'Computer Name'")
56 computerRole=$(/usr/libexec/plistbuddy /var/tmp/DEPNotify.plist -c "print 'Computer Role'")
57
58 # Update Hostname and Computer Role in JSS
59 # Create xml
60 cat << EOF > /var/tmp/name.xml
61 <computer>
62     <extension_attributes>
63         <extension_attribute>
64             <name>Hostname</name>
65             <value>$computerName</value>
66         </extension_attribute>
67     </extension_attributes>
68 </computer>
69 EOF
70 ## Upload the xml file
71 /usr/bin/curl -sfku "$5"
72 # Create xml
73 cat << EOF > /var/tmp/role
74 <computer>
75     <extension_attributes>
76         <extension_attribute>
77             <name>Mac User Role</name>
78             <value>$computerRole</value>
79         </extension_attribute>
```



The screenshot shows a Mac OS X window titled "Hi there!" from UEL. The window contains the following text:

It's time to set up this Mac with the software and settings it needs. Before we continue, please make sure it is plugged into a wired network connection on campus.

If you need any assistance, please contact the UEL IT Service Desk.

Telephone: 020 8223 2558
Email: servicedesk@uel.ac.uk

Please set the computer name and role to continue...

Continues

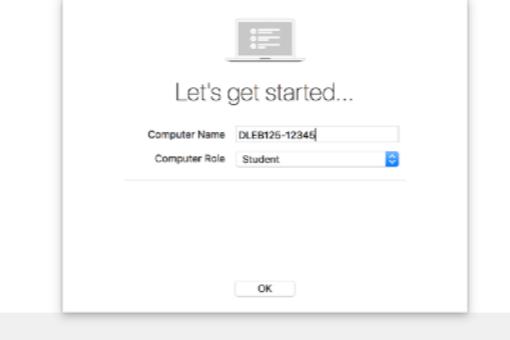
Now we launch DEPNotify as the logged in user, along with the Jamf and fullscreen switches.

Its window appears and the surrounding screen area is blurred.

It will read the Jamf log and update the status text when policies run and packages are installed.

We click the Continue button

```
47
48 # Wait for the user data to be submitted...
49 while [ ! -f /var/tmp/DEPNotify.plist ]; do
50     echo "Status: Please set the computer name and role to continue..." >> /var/tmp/depnotify.log
51     sleep 5
52 done
53
54 # Let's read the user data into some variables...
55 computerName=$(/usr/libexec/plistbuddy /var/tmp/DEPNotify.plist -c "print 'Computer Name'")
56 computerRole=$(/usr/libexec/plistbuddy /var/tmp/DEPNotify.plist -c "print 'Computer Role'")
57
58 # Update Hostname and Computer Role in JSS
59 # Create xml
60 cat << EOF > /var/tmp/name.xml
61 <computer>
62     <extension_attributes>
63         <extension_attribute>
64             <name>Hostname</name>
65             <value>$computerName</value>
66         </extension_attribute>
67     </extension_attributes>
68 </computer>
69 EOF
70 ## Upload the xml file
71 /usr/bin/curl -sfku "$5":"
72 # Create xml
73 cat << EOF > /var/tmp/role
74 <computer>
75     <extension_attributes>
76         <extension_attribute>
77             <name>Mac User Role</name>
78             <value>$computerRole</value>
79         </extension_attribute>
80     </extension_attributes>
81 </computer>
82 EOF
```

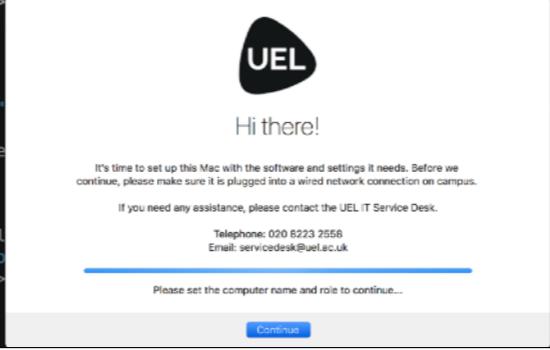


```
curl -T /var/tmp/name.xml -X PUT
```

We won't continue until the plist containing our user input data is written.

Our technician enters the hostname and chooses the role, then click OK.

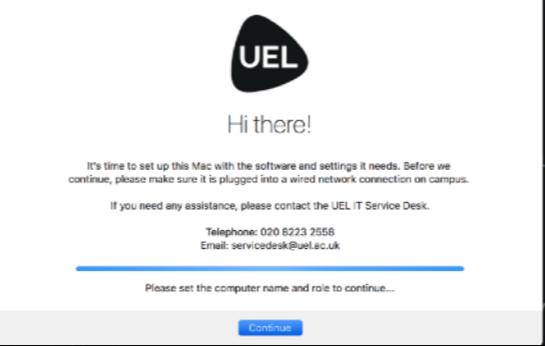
```
47
48 # Wait for the user data to be submitted...
49 while [ ! -f /var/tmp/DEPNotify.plist ]; do
50     echo "Status: Please set the computer name and role to continue..." >> /var/tmp/depnotify.log
51     sleep 5
52 done
53
54 # Let's read the user data into some variables...
55 computerName=$(/usr/libexec/plistbuddy /var/tmp/DEPNotify.plist -c "print 'Computer Name'")
56 computerRole=$(/usr/libexec/plistbuddy /var/tmp/DEPNotify.plist -c "print 'Computer Role'")
57
58 # Update Hostname and Computer Role in JSS
59 # Create xml
60 cat << EOF > /var/tmp/name.xml
61 <computer>
62     <extension_attributes>
63         <extension_attribute>
64             <name>Hostname</name>
65             <value>$computerName</value>
66         </extension_attribute>
67     </extension_attributes>
68 </computer>
69 EOF
70 ## Upload the xml file
71 /usr/bin/curl -sfku "$5":"
72 # Create xml
73 cat << EOF > /var/tmp/role
74 <computer>
75     <extension_attributes>
76         <extension_attribute>
77             <name>Mac User Role</name>
78             <value>$computerRole</value>
79         </extension_attribute>
80     </extension_attributes>
81 </computer>
82 EOF
83
```



The image shows a screenshot of a web-based setup interface for a UEL Mac. At the top right is the UEL logo. Below it, a message says 'Hi there!'. A central text area contains instructions: 'It's time to set up this Mac with the software and settings it needs. Before we continue, please make sure it is plugged into a wired network connection on campus.' It also provides contact information: 'If you need any assistance, please contact the UEL IT Service Desk. Telephone: 020 8223 2558 Email: servicedesk@uel.ac.uk'. At the bottom, a text input field says 'Please set the computer name and role to continue...' with a blue 'Continue' button.

DEPNotify writes the plist and we read back the hostname and role into a couple of variables.

```
58      # Update Hostname and Computer Role in JSS
59      # Create xml
60      cat << EOF > /var/tmp/name.xml
61      <computer>
62          <extension_attributes>
63              <extension_attribute>
64                  <name>Hostname</name>
65                  <value>$computerName</value>
66              </extension_attribute>
67          </extension_attributes>
68      </computer>
69 EOF
70      ## Upload the xml file
71      /usr/bin/curl -sfku "$5":"$6" "$4":8443/JSSResource/computers/serialnumber/"$serial" -T /var/tmp/name.xml -X PUT
72      # Create XML
73      cat << EOF > /var/tmp/role.xml
74      <computer>
75          <extension_attributes>
76              <extension_attribute>
77                  <name>Mac User Role</name>
78                  <value>$computerRole</value>
79              </extension_attribute>
80          </extension_attributes>
81      </computer>
82 EOF
83      ## Upload the xml file
84      /usr/bin/curl -sfku "$5":"$6" "$4":8443/JSSResource/computers/serialnumber/"$serial" -T /var/tmp/role.xml -X PUT
85
86 else
87     # Set variables for Computer
88     computerName="$jssHostName"
89     computerRole="$jssUserRole"
90     # Launch DEPNotify
91     echo "Command: Image: "/Library/Application Support/DEPNotify/DEPNotify.app/Contents/MacOS/DEPNotify
92     echo "Command: MainTitle: UEL"
```



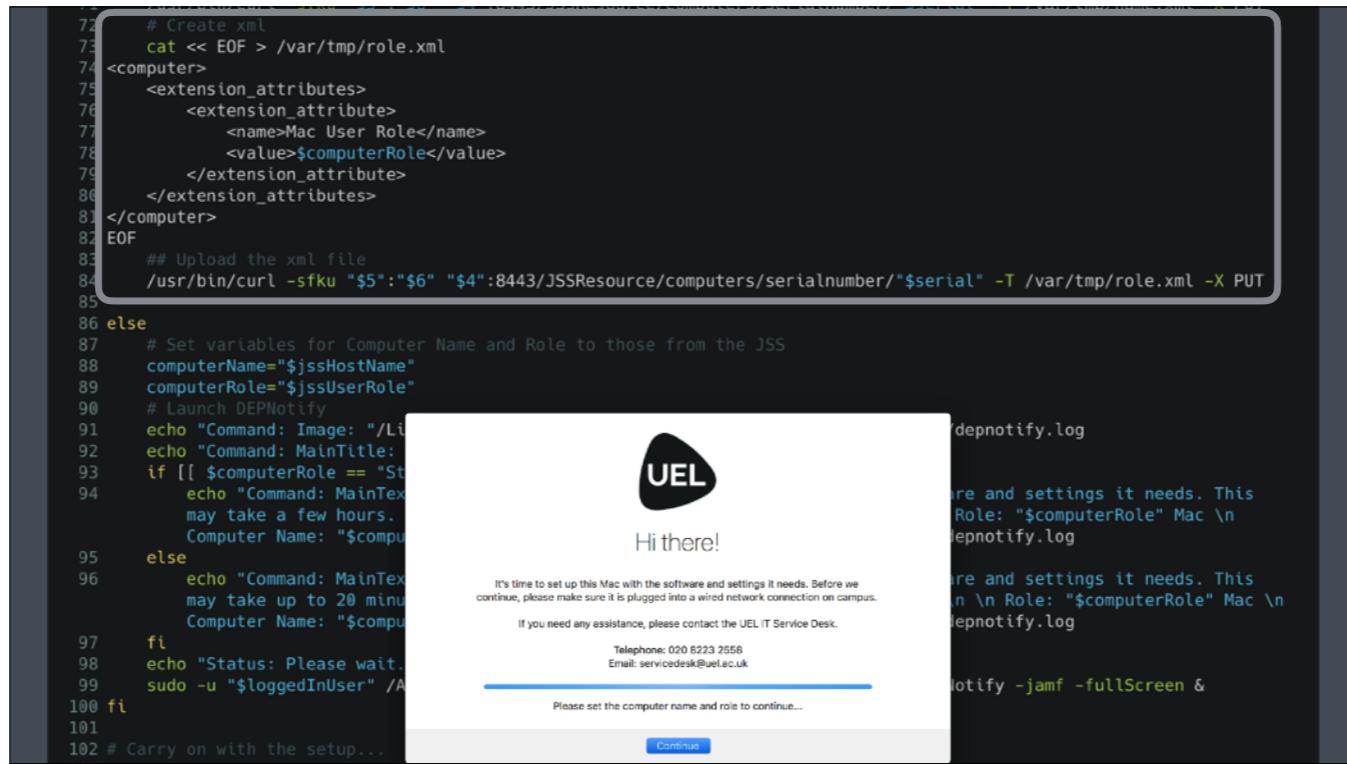
The screenshot shows a web-based setup interface for a Mac. At the top is a dark header bar with the UEL logo. Below it is a white form area with the following content:

- A large black button with the text "Hi there!"
- A message: "It's time to set up this Mac with the software and settings it needs. Before we continue, please make sure it is plugged into a wired network connection on campus."
- A note: "If you need any assistance, please contact the UEL IT Service Desk." followed by contact details: "Telephone: 020 8223 2558" and "Email: servicedesk@uel.ac.uk".
- A blue horizontal bar with the text "Please set the computer name and role to continue...".
- A small blue "Continue" button at the bottom right.

To the right of the form, the terminal window shows the command being run: "al" -T /var/tmp/role.xml -X PUT. Below the terminal window, the file "depnotify.log" is visible.

Next, we create some XML containing the name and use the API to update the Computer Name Extension Attribute with it.

```
72     # Create xml
73     cat << EOF > /var/tmp/role.xml
74 <computer>
75     <extension_attributes>
76         <extension_attribute>
77             <name>Mac User Role</name>
78             <value>$computerRole</value>
79         </extension_attribute>
80     </extension_attributes>
81 </computer>
82 EOF
83     ## Upload the xml file
84     /usr/bin/curl -sfku "$5":"$6" "$4":8443/JSSResource/computers/serialnumber/"$serial" -T /var/tmp/role.xml -X PUT
85
86 else
87     # Set variables for Computer Name and Role to those from the JSS
88     computerName="$jssHostName"
89     computerRole="$jssUserRole"
90     # Launch DEPNotify
91     echo "Command: Image: /Li
92     echo "Command: MainTitle:
93     if [[ $computerRole == "St
94         echo "Command: MainText:
95         may take a few hours.
96         Computer Name: "$computerName"
97     else
98         echo "Command: MainText:
99         may take up to 20 minutes.
100        Computer Name: "$computerName"
101    fi
102 echo "Status: Please wait.
103 sudo -u "$loggedInUser" /A
104 fi
105
106 # Carry on with the setup...
107
```



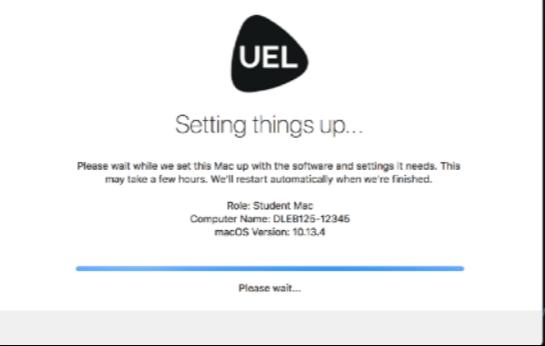
The terminal window shows the script being run. A modal dialog box from UEL IT Service Desk appears, prompting the user to set the computer name and role. The dialog includes the UEL logo, a greeting, instructions, contact information, and a 'Continue' button.

We do the same for the role.

```
72 # Create xml
73 cat << EOF > /var/tmp/role.xml
74 <computer>
75   <extension_attributes>
76     <extension_attribute>
77       <name>Mac User Role</name>
78       <value>$computerRole</value>
79     </extension_attribute>
80   </extension_attributes>
81 </computer>
82 EOF
83 ## Upload the xml file
84 /usr/bin/curl -sfku "$5":"$6" "$4":8443/JSSResource/computers/serialnumber/"$serial" -T /var/tmp/role.xml -X PUT
85
86 else
87   # Set variables for Computer Name and Role to those from the JSS
88   computerName="$jssHostName"
89   computerRole="$jssUserRole"
90   # Launch DEPNotify
91   echo "Command: Image:=\"/Library/Application Support/UEL/ux/UEL.png\" >> /var/tmp/depnotify.log"
92   echo "Command: MainTitle: Setting things up..." >> /var/tmp/depnotify.log
93   if [[ $computerRole == "Student" ]]; then
94     echo "Command: MainText: Please wait while we set this Mac up with the software and settings it needs. This
95     may take a few hours. We'll restart automatically when we're finished. \n \n Role: \"$computerRole\" Mac \n
96     Computer Name: \"$computerName\" \n macOS Version: \"$osversion\" >> /var/tmp/depnotify.log
97   else
98     echo "Command: MainText: Please wait while we set this Mac up with the software and settings it needs. This
99     may take up to 20 minutes. We'll restart automatically when we're finished. \n \n Role: \"$computerRole\" Mac \n
100    Computer Name: \"$computerName\" \n macOS Version: \"$osversion\" >> /var/tmp/depnotify.log
101  fi
102 echo "Status: Please wait..." >> /var/tmp/depnotify.log
103 sudo -u "$loggedInUser" /Applications/Utilities/DEPNotify.app/Contents/MacOS/DEPNotify -jamf -fullScreen &
104
105 # Carry on with the setup...
106
```

Now, if the name and role are already set, we'd skip all of that - so we're not prompted if we're re-provisioning a Mac that's already in Jamf.

```
102 # Carry on with the setup...
103
104 # Change DEPNotify title and text...
105 echo "Command: MainTitle: Setting things up..." >> /var/tmp/deponotify.log
106 if [[ $computerRole == "Student" ]]; then
107   echo "Command: MainText: Please wait while we set this Mac up with the software and settings it needs. This may
take a few hours. We'll restart automatically when we're finished. \n \n Role: \"$computerRole" Mac \n Computer
Name: \"$computerName" \n macOS Version: \"$osversion"" >> /var/tmp/deponotify.log
108 else
109   echo "Command: MainText: Please wait while we set this Mac up with the software and settings it needs. This may
take up to 20 minutes. We'll restart automatically when we're finished. \n \n Role: \"$computerRole" Mac \n
Computer Name: \"$computerName" \n macOS Version: \"$osversion"" >> /var/tmp/deponotify.log
110 fi
111 echo "Status: Please wait..." >> /var/tmp/deponotify.log
112
113 # Time to set the hostname...
114 echo "Status: Setting computer name" >> /var/tmp/deponotify.log
115 jamf setComputerName -name "${computerName}"
116
117 # Bind to AD
118 jamf policy -event BindAD
119
120 # Run software deployment policy
121 jamf policy -event Deploy
122
123 # Run a software update
124
125 echo "Status: Installing Apple Software Update"
126 /usr/sbin/softwareupdate -ia
127
128 # Finishing up
129 echo "Command: MainTitle: All setup complete"
130 echo "Command: MainText: This please contact the UEL IT Services team if you need any assistance."
131 echo "Status: Please wait..." >> /var/tmp/deponotify.log
```

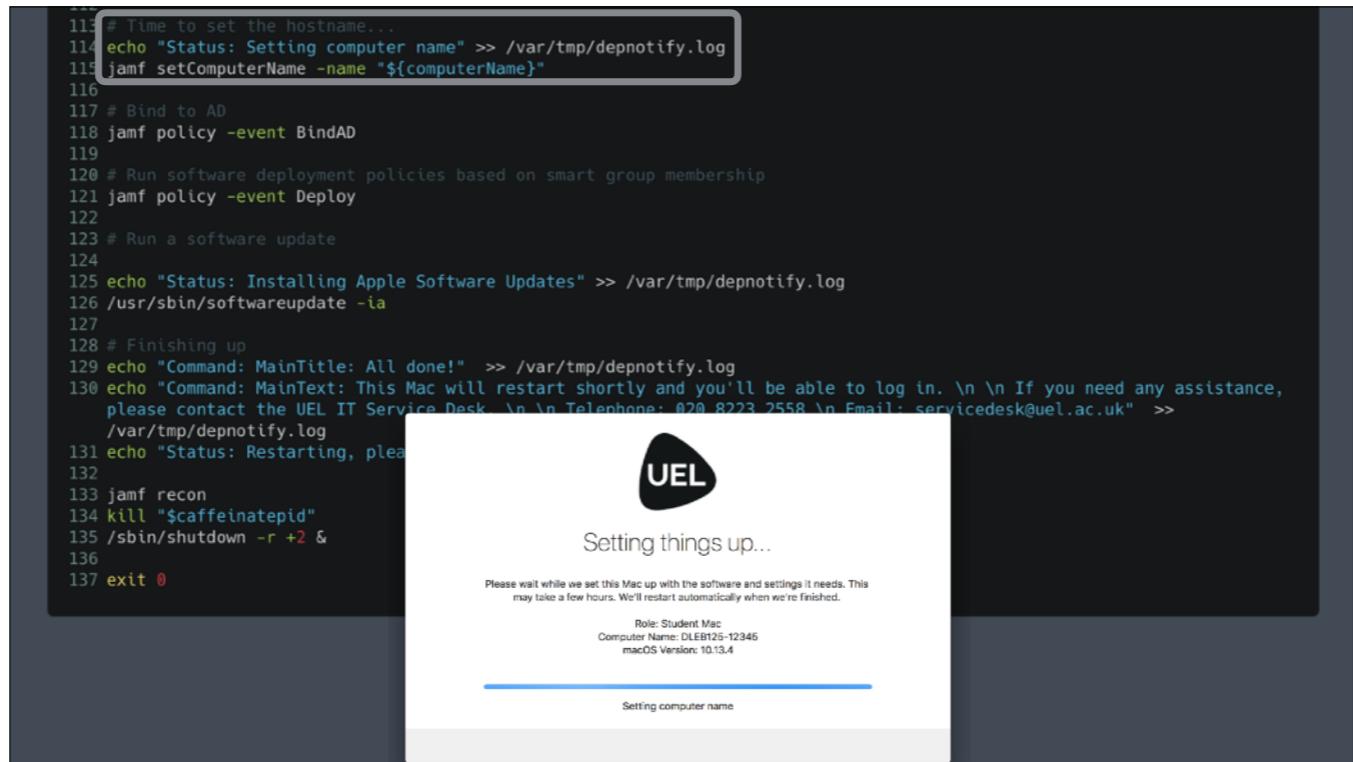


The screenshot shows a DEPNotify window titled "Setting things up...". The main text reads: "Please wait while we set this Mac up with the software and settings it needs. This may take a few hours. We'll restart automatically when we're finished." Below this, it specifies the computer's role as "Student Mac", name as "DLEB125-12345", and macOS version as "macOS Version: 10.13.4". At the bottom, there is a progress bar labeled "Please wait..." and a link "If you need any assistance, email icedesk@uel.ac.uk".

We carry on - changing the title, main text and status to let us know what's happening.

We can also include the hostname and version of macOS here.

If it's a student facing Mac, our main text lets us know it'll take longer to set up than a staff facing Mac.



We set the hostname

```
113 # Time to set the hostname...
114 echo "Status: Setting computer name" >> /var/tmp/depnotify.log
115 jamf setComputerName -name "${computerName}"
116
117 # Bind to AD
118 jamf policy -event BindAD
119
120 # Run software deployment policies based on smart group membership
121 jamf policy -event Deploy
122
123 # Run a software update
124
125 echo "Status: Installing Apple Software Updates" >> /var/tmp/depnotify.log
126 /usr/sbin/softwareupdate -ia
127
128 # Finishing up
129 echo "Command: MainTitle: All done!" >> /var/tmp/depnotify.log
130 echo "Command: MainText: This Mac will restart shortly and you'll be able to log in. \n \n If you need any assistance,
    please contact the UEL IT Service Desk. \n \n Telephone: 020 8223 2558 \n Email: servicedesk@uel.ac.uk" >>
    /var/tmp/depnotify.log
131 echo "Status: Restarting, plea
132
133 jamf recon
134 kill "$caffeinatepid"
135 /sbin/shutdown -r +2 &
136
137 exit 0
```



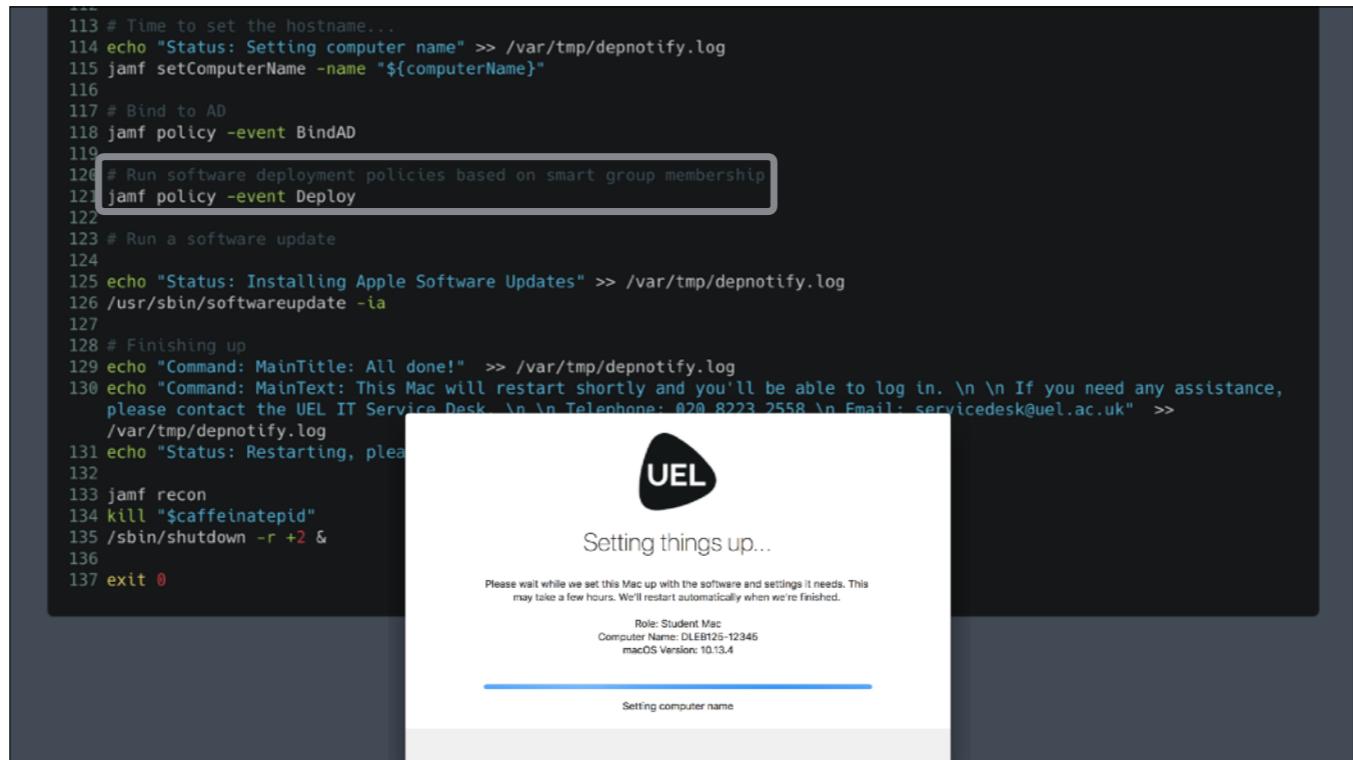
Setting things up...

Please wait while we set this Mac up with the software and settings it needs. This may take a few hours. We'll restart automatically when we're finished.

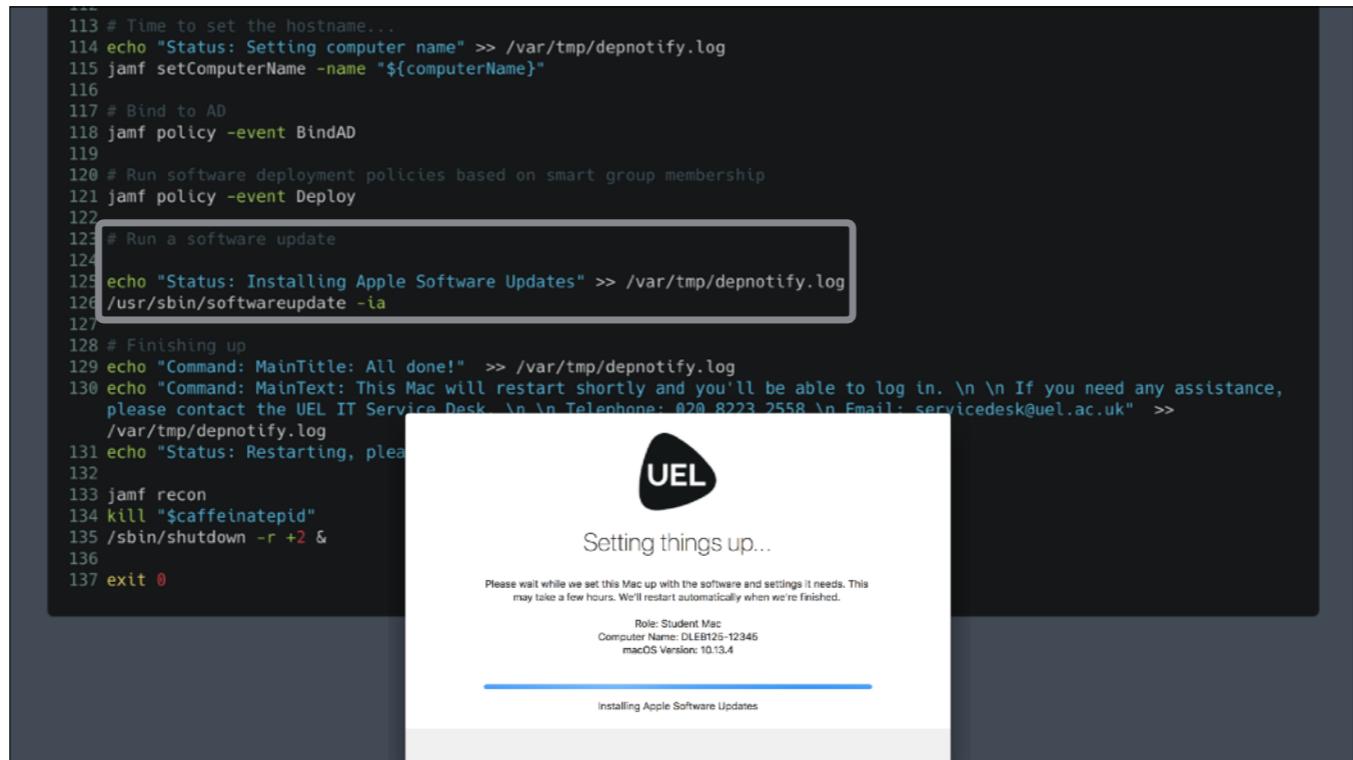
Role: Student Mac
Computer Name: DLEB125-12345
macOS Version: 10.13.4

Setting computer name

Bind to Active Directory

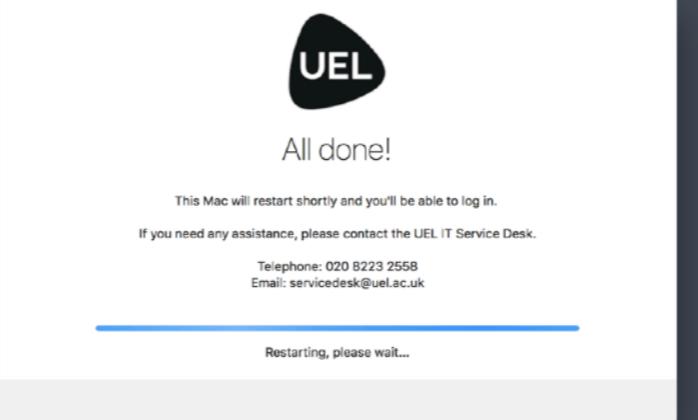


And run any custom policy triggers we like. Our Mac will be in the right lab smart group and the relevant policies to install that lab's specific software will be scoped to that smart group. Because of that, those policies can all have the same custom trigger.



We run a software update - that's important because when you install a clean version of macOS, it doesn't include security updates or other updates like iTunes and Safari, like an image can.

```
127
128 # Finishing up
129 echo "Command: MainTitle: All done!" >> /var/tmp/depnotify.log
130 echo "Command: MainText: This Mac will restart shortly and you'll be able to log in. \n \n If you need any assistance,
131   please contact the UEL IT Service Desk. \n \n Telephone: 020 8223 2558 \n Email: servicedesk@uel.ac.uk" >>
132   /var/tmp/depnotify.log
133 echo "Status: Restarting, please wait..." >> /var/tmp/depnotify.log
134
135 jamf recon
136 kill "$caffeinatepid"
137 /sbin/shutdown -r +2 &
138
139 exit 0
```



The image shows a dark grey background with a central white rectangular box. At the top center is a black rounded triangle containing the letters 'UEL'. Below it, the text 'All done!' is centered. Underneath that, smaller text reads: 'This Mac will restart shortly and you'll be able to log in. If you need any assistance, please contact the UEL IT Service Desk. Telephone: 020 8223 2558 Email: servicedesk@uel.ac.uk'. A thin horizontal blue line is at the bottom, followed by the text 'Restarting, please wait...'.

And we're finished, so we tell DEPNotify to let us know, update inventory and restart with a 2 minute delay - so the policy has time to complete and write its log back to Jamf.

The future

- Skip the Setup Assistant entirely
- Do it without logging in
- Asset management DB integration
- Apple don't sit still



So, I said at the beginning that this was a journey I was TAKING. There are lots of places we could continue to take this to.

<C> This is not zero touch - it's as close as I can get it, but maybe Apple could include an option to skip the entire Setup Assistant

<C> Do we really need to log in? At the moment, yes. DEPNotify won't run on top of the Login Window. But that might change and there are some exciting conversations happening...

<C> Instead of storing and reading the hostname and role from within Jamf, we could point to an asset management tool and store that data outside. That means we could delete the computer record in Jamf before we start, but still skip asking for the data. I do sometimes notice issues like profiles not pushing properly when we re-provision a Mac with an existing computer record. - this might improve things.

<C> Apple change stuff. Our workflows might break tomorrow. We can only do what we can. And because I called this talk "Dawn of the DEP"...



It might be best to go to the Winchester, have a nice cold pint, and wait for all this to blow over.



Thank you

Resources

<https://soundmacguy.wordpress.com/2018/05/17/lab-nauseam-dawn-of-the-dep/>

