





jamf

NATION


User Conference

2018



## David Ramirez

Senior Service Manager



© JAMF Software, LLC

My name is David Ramirez and I am Senior Service Manager for MacOS and Jamf Services. I'm extremely excited for this opportunity. This this is my first time presenting here at JNUC. I want to give huge shoutout to the PDX macadmins community for their support. They were first to see me present this and test out the tool externally.



## Simplifying User Interaction

Presentation agenda:

What is the tool?

Why does it exist?

What can it do?

How does it work?

Where do I get it?

How do I set it up myself?



© JAMF Software, LLC

Today we're going to talk about and share a tool that I've created which simplifies user interaction using jamf policies. So this is what I'll go over with you.

## A Beginner's Tale - 2014

Mac Desktop Support  
Only wrote small  
scripts in AppleScript  
Volunteered to co-  
create the UEX toolkit  
Strong Google-fu



In 2014 in Portland Oregon I was working as a Deskside supporter and I had only started out playing with AppleScript. I spent a lot of time reading and trying to understand the scripts that our Jamf admin had wrote in hopes that one day I could do what he did.

About 6 months in, there was a demand from our managers to create a standard toolkit for interacting with the user. They wanted to allow them to interact with policies and inform them about restarts or quitting applications.

The Jamf admin had his plate completely full. Foolhardily I volunteered to work with another great guy to build it. To me, my colleague is a “scripting greybeard”. But before we could really get started he found another opportunity and left....

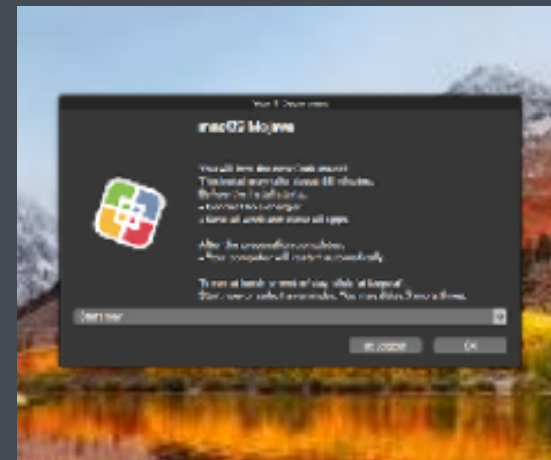
I had lost my sensei.

\*

So I had to become really strong with my google-fu.

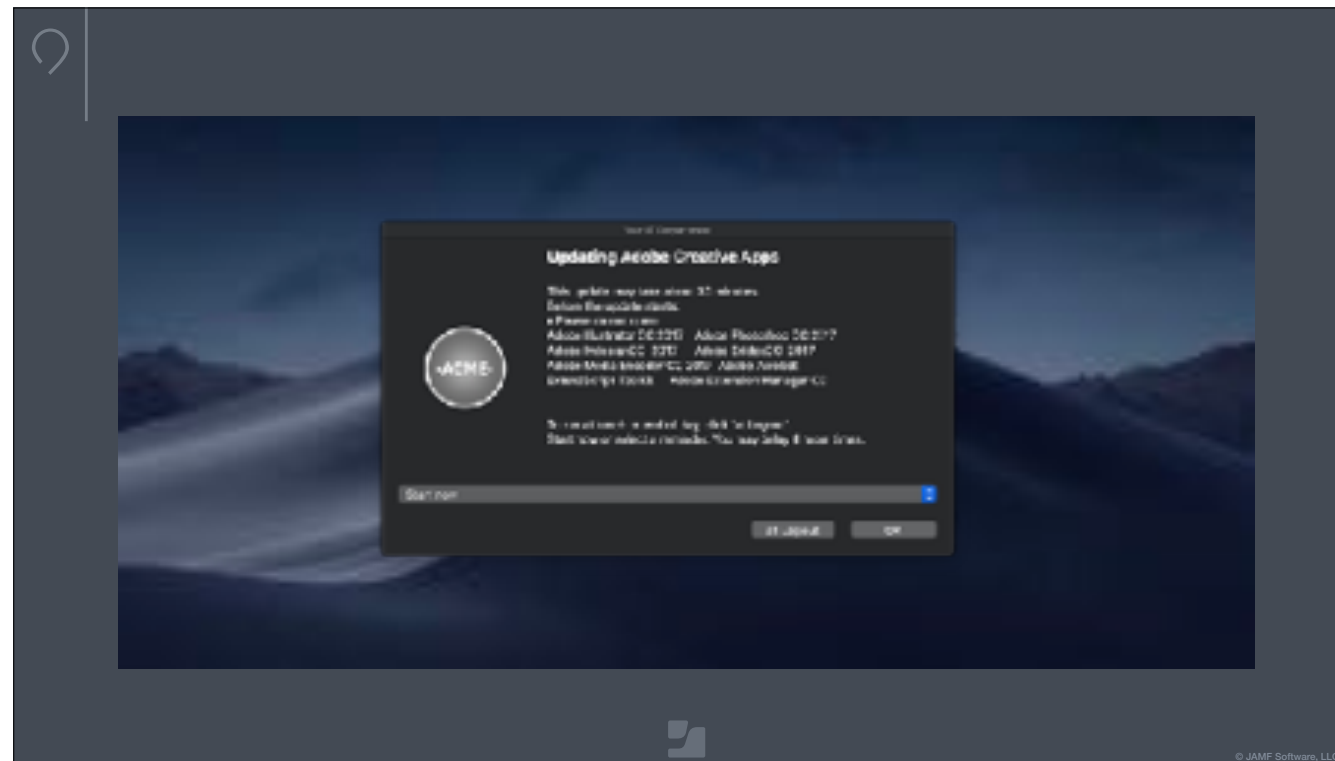
## Jamf-Interaction-Toolkit

Standardized UEX  
Toolkit to display  
dialog and support  
postponing running  
policies.



It's been over three years now since our first build and now we have a tool we internally call UEX.

The Jamf Interaction toolkit is now a very versatile set of scripts that is our standard way we interact with users.



Here are a few examples of what we use it for.

## Why does it exist?

Custom branding  
Only when needed  
Built in delay function  
Works well with jamf  
Easy to comprehend  
Easy to use and repeat



© JAMF Software, LLC

At every JNUC these are some of the common things recommended when setting up on screen interaction with your users.

- \*custom branding
- \*only display dialogs when they're necessary
- \*built in delay
- \*compatible with native jamf functions
  - \* and simplicity for the user and admin
  - \* UEX can do all of this and a ton more. Lest watch a demo of two ways its used.



In this example we have a policy running at check in to update Spotify to a new version but the user has the application running and listening to their guilty pleasure song. Let s see what happens...

- They decide Nows not a good time and choose to delays the installation for 1 day
- then they get a reminder that can do it from Self Service any time
- So they're done with their "guilty pleasure" playlist quit the app and now go to Self service to update it
  - From self service they get status dialogs when the install starts and finishes
  - Then they can open the app again an jam out to another fun tune
  -



## What can it do?

### Primary functions

- QUIT
- BLOCK
- LOGOUT
- RESTART

- So what can it do
- \*Quitting apps is the most basic which will inform the user to quit the app if it's running
- \*Block expand on quit and can be used for long installations where certain applications shouldn't be opened while the install is running
- \*Logout will log the user out after an install completes
- \*Restart will restart the computer with a friendly branded dialog after the installation completes

## What can it do?

### Built in delay support

- User chosen delay
- Inactivity delay
- Presentation delay

There are three types of delays that are built in to every interaction

- \*The main bread and butter is allowing the user to choose a delay period that is limited by a number of delays
- \*Then if the the user is not at their computer when the policy runs it tries every two hours up to three times and then takes away a delay.
- \*The last delay option will automatically suppress popup up to 3 times if the is running
  - Power Point
  - Keynote
  - Vidyo Desktop
  - In a Webex Meeting
  - Or People + Content IP

Lets see an example of of the user delay works.



This time the user is browsing the jamf market place and then gets a software install request requiring a restart. They choose a delay of 1 hour.

So lets see what happens.

You can see that the user can no longer postpone the installation. They start the install and get the notifications Which is completely up to you. So once the install complete the user is prompted to restart. How do I do this? I follow the philosophy of Bruce.

9

## Parameters are friends

...“not food”

Script parameters in a jamf policy are a simple way to do more using a single script without making a new script each time.

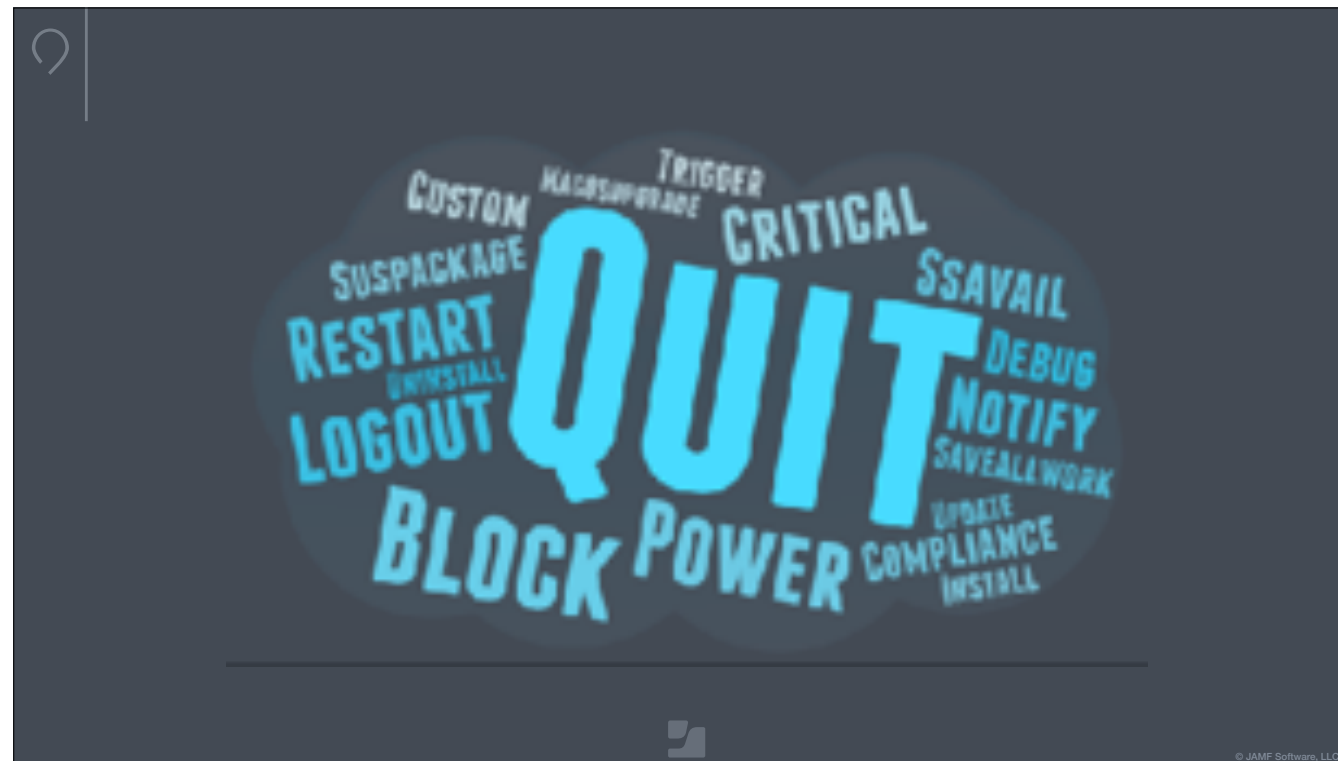


© JAMF Software, LLC

“Parameters are friends...NOT FOOD”

\*

Jamf has a powerful feature that can feed up to 7 parameters to your script. I’ll show you how it’s implemented later



These are all of the checks for how you can change the interaction experience and install flow. So... here are three more quick examples of new advanced features built in to the tool



\*

Now if an app is being quit the user can either quit it on their own and the installation will start right away then the app will be re-opened after the install completes.

Also it will not try a safe quit to prompt saving the document then the app re opens

And also if the users or cancel the quit the user will get reminder and it will kill the app then the app re opens



Here you'll see that if the user has already approved a restart of their computer they don't get asked again unless there is some other requirement like power or quitting an app. This also works with logout.



Here you'll see that there is requirement to quit application for the first install of the new Citrix Workspace application. Then say later you're doing an update of that application and the dialog is changed dynamically to inform the user that it's an update instead of a first time install All with the same policy and no changes needed by the admin

So now I'm going to show you how it works from the jamf Pro server



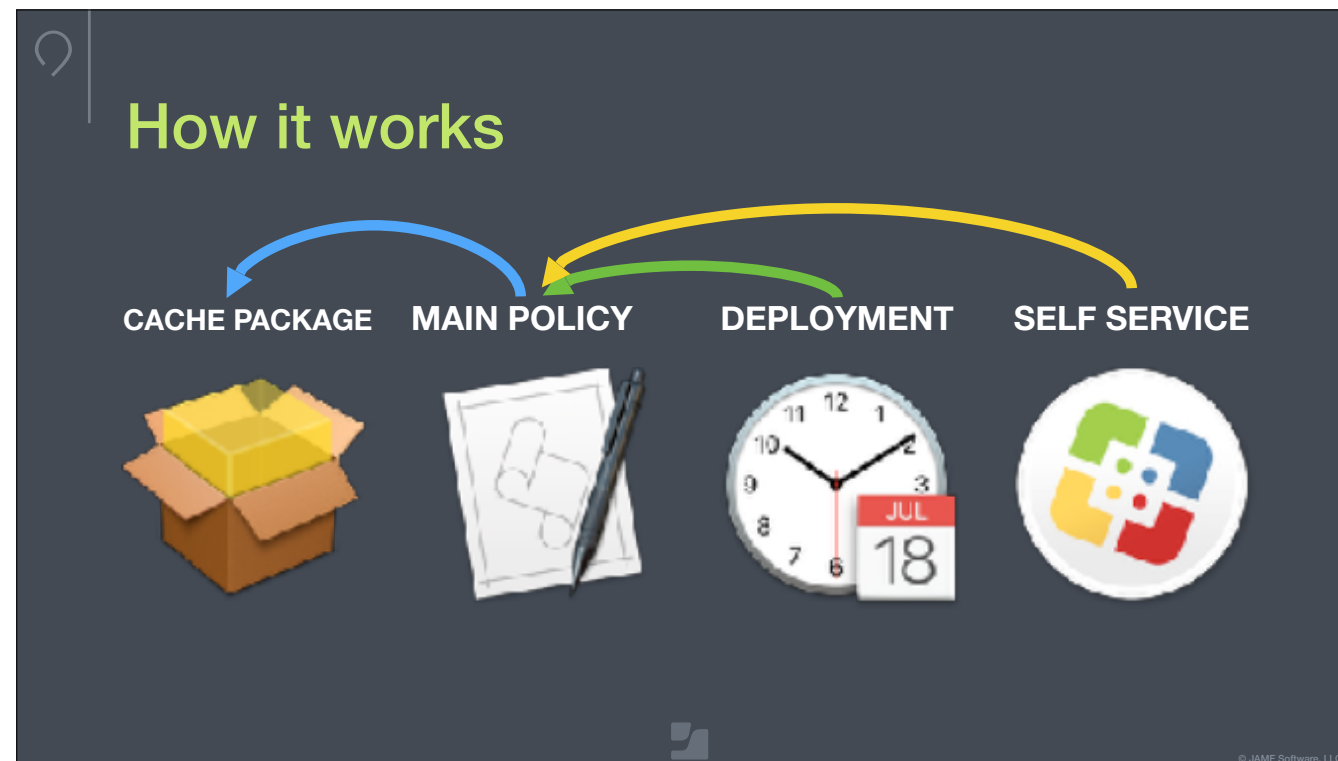
## How it works

MAIN POLICY



© JAMF Software, LLC

Let's start with the high level structure.  
It all starts with one main policy that uses a custom event to trigger



From there you create another policy to instruct the computer to cache the package. This allows the script to determine if the package needs to be cached or if it's already there.

\*Then you create your deployment policy with whatever scope need. \*But instead re-doing all the scripts parameter you simply refer to the main policy like a "recipe".

\*You can do the same thing in another policy for self service to get behaviour more appropriate for self service.

Now I'll show you what it looks like on the jamf pro server.



So on the Jamf Pro server we create the policies with the UEX recipe and the policy to cache the package. You use the same custom event name for both but add `_cache` to the package one.

\*

You'll see that the UEX Recipe runs the script `UEX-Jamf-Interaction` with all the parameters. I add the "no Grep" part in the name so other scripts in package don't kill this process when they're running. I was going to call it "don't grep me Bro" but this cleaner.

The package caching policy has the action to cache the package.

The UEX Script know If the package is already cached to not download it again.

\*

Let me explain the parameters. The first one is the vendor, app name and version. I have another feature I'm working on that you can add the space required but that's not ready yet.

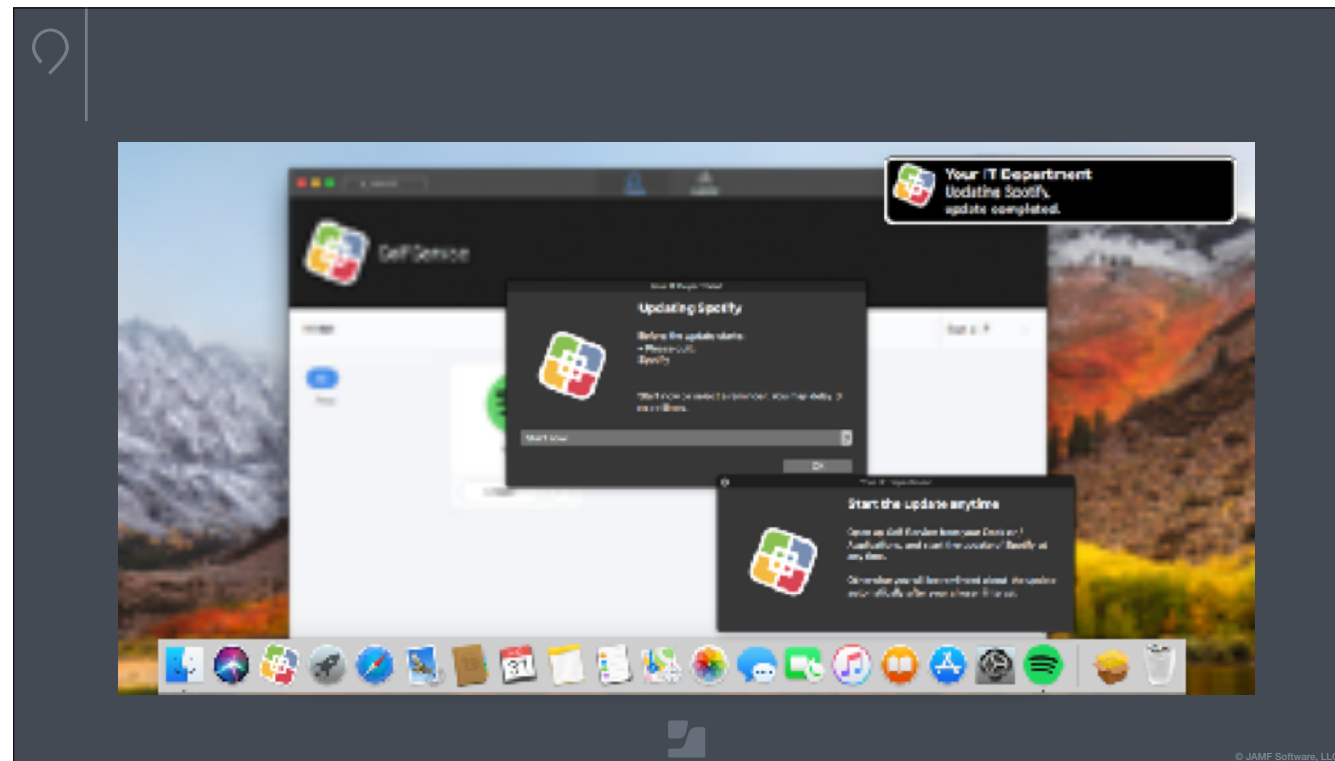
Then you have a check to quit the app and that it's available in SS. Also that it's an update

Then you specify that app that needs to be quit

how long the install is

How many times the user can postpone the install

The filename of the package that is cached, You can also add multiple packages here and separate them



So if the policy is running from self service the user get a cancel and start now button. But only if Spotify is running.\*

If they start it it will tell them that it's downloading the package\* then when it's starting and completing the update\*

When a policy is run at check in then the use is interrupted from their work so it offers the user a chance to delay the actions for 1-4 hours or until tomorrow.\* If they choose to post pone it then it will tell them that it's available in self service to run at any time. Fun fact if you've named self service differently it will change that name and also tell you where it can found if it's not in the dock.

## How the deferral works

- Extension attribute looks in the deferral folder and returns active if there are plists in there.
- Smart group collects all the active ones
- On every check-in, computers in the group run a policy to check if enough time has passed then trigger the UEX policy



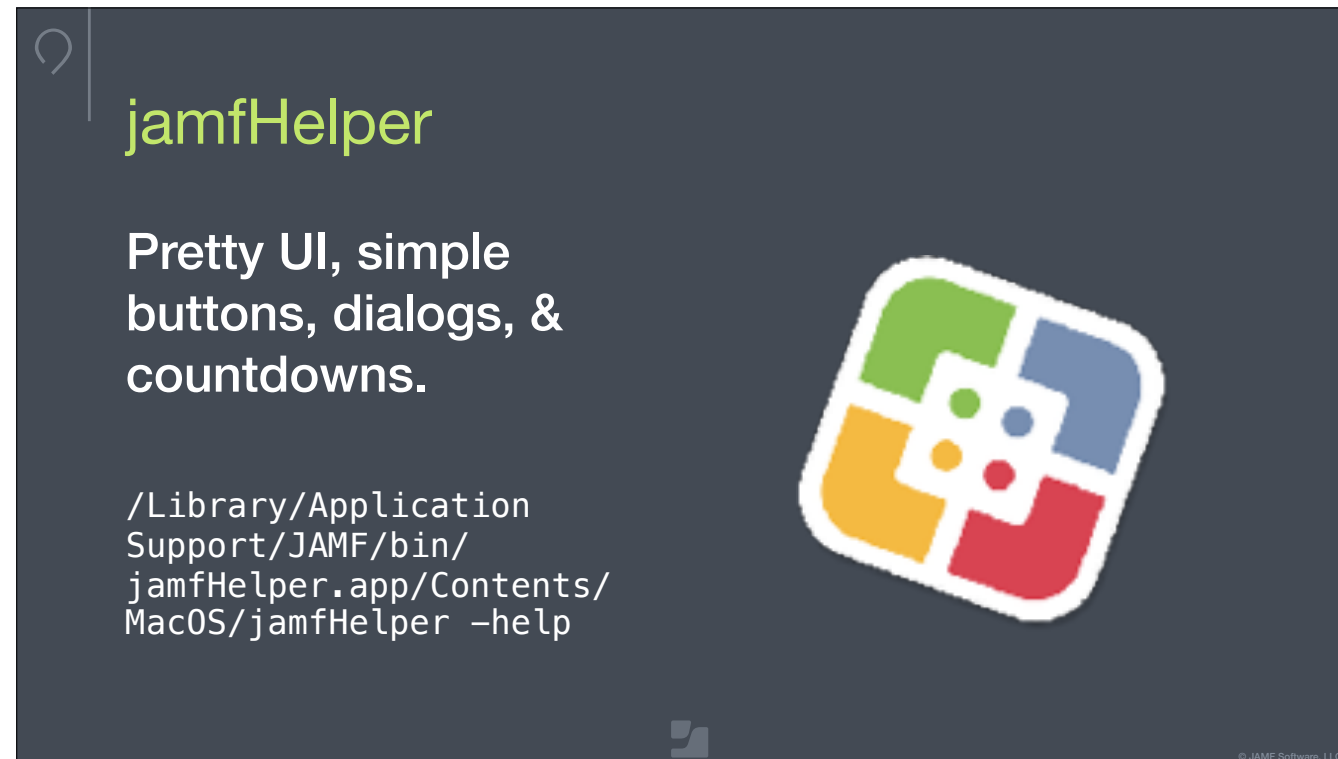
© JAMF Software, LLC

So lets go over a little on how the deferral system works

\*Extension attributes looks in the deferral folder and returns active if there are plists in there.

\*Smart group collects all the active ones

\* On every check-in, computers in the group run a policy to check if enough time has passed then trigger the UEX policy



Here are the UI Tool I use to make all the dialogs

jamfHelper is on every Mac Manage by jamf and has many uses. Including shooing deferral options.

You just run that command there and you can see all the options that available.

## cocoaDialog - markcarver - mstratman

Create macOS dialogs  
from the command line  
easily!

[https://  
cocoadialog.com](https://cocoadialog.com)



© JAMF Software, LLC

Currently I'm using v 2.8 of CocoaDialog. I like for the very elegant bubble notifications.

## Please Wait - macbrained

Clean loading bar that  
can dynamically  
change.

[https://github.com/  
macbrained/User-  
Interfaces](https://github.com/macbrained/User-Interfaces)



© JAMF Software, LLC

This makes nice a little progress bar that is used to commentate the. Progress of really long installations.



## Helper Agent Policies

- PLEASEWAITUPDATER
- UEXDEFERRALSERVICE
- UEXBLOCKAGENT
- UEXLOGOUTAGENT
- UEXRESTARTAGENT
- UEX\_INVENTORY\_UPDATE\_AGENT

These are the event names for other policies that are used in conjunction with the master script make the whole experience possible.

PleaseWaitUpdater

Starts the progress bar indicator for long installs and display the requirements of the UEX Policy.

UEXDeferralservice

This checks the system for existing UEX policies that have been deferred by the user and runs them when enough time has passed

UEXBLOCKAGENT

This agent blocks application from if required by the UEX policy.

UEXlogoutagent

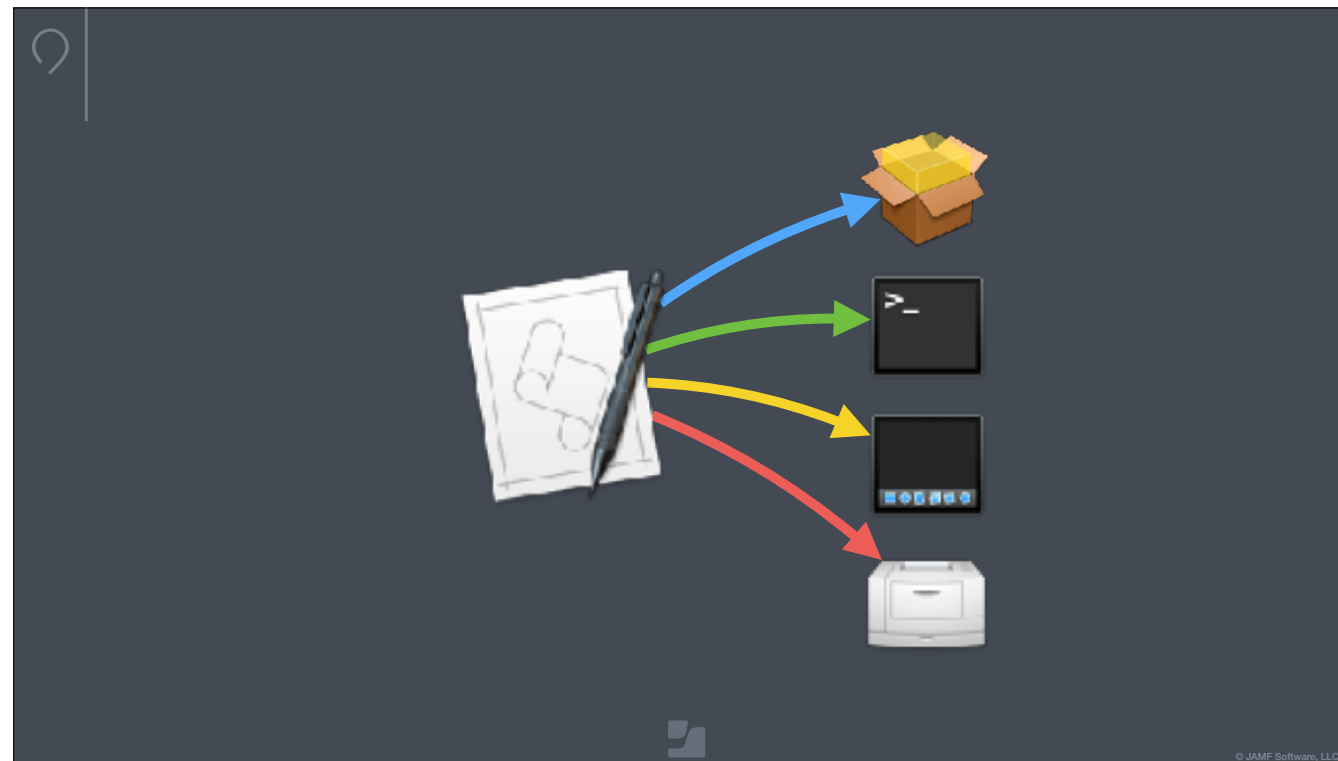
This agent prompts then logs the user out of their account after all policies have completed.

UEXREstartagent

This agent prompts then restarts the computer after all policies have completed

uex\_inventory\_update\_agent

Used to update the inventory when the deferral status changes or when the install has ben run.



By linking it to another policy using the trigger parameter you can make it do all sorts of stuff. Like other scrips dock times or printers



A more complex example would be one for macOS Mojave upgrades

Here I use the checks macOSUpgrade to change the wording of the install window, ssavail since the user can also use self service

Trigger to run another policy with the installation script

Save all work to remind the user to do just that

And power to say that power is required in order to install this. There is a built in check that won't proceed until power is connected.

Then in the triggers parameter I put the UEX recipe event name and then the event name for the policy that has the work flow of updating the OS

\*

Then this what the deployment version of the UEX install would look like The run at logout option just records that the user's approved the install to run at the login window and is triggered by a logout

\*

When the user chooses to run it then it will trigger the other policies you've given.

## Where do I get it?

[https://github.com/  
cubandave/Jamf-  
Interaction-Toolkit](https://github.com/cubandave/Jamf-Interaction-Toolkit)

#uex-tool-for-jamf on  
Slack



So I hope I know what you're thinking now. Where the heck to I get this this?

Its available for free on GitHub with an Apache license meaning use at your risk and also please give us credit if you fork and modify it.

There I'm keeping track of future feature that I want to developer and the full wiki and issue tracker are there.

We also have a channel on the macAdmin slack where you're welcome to start conversation and ask questions.



## How do I set it up myself?

### Uploading Components

- 1 main script
- 6 helper scripts + 1 package with tools
- 1 extension attribute
- Run Setup tool



© JAMF Software, LLC

Now the fun bit is I'll show you in less than 5 mins how you can configure your jamf pro server to do the same thing



## Updating Company Name

00-UEX-Jamf-  
Interaction-no-grep

00-uexblockagent-jss  
00-uexlogoutagent-jss  
00-uexrestartagent-jss



© JAMF Software, LLC

If you want to personalise the name of your IT Org you just update these files if you add custom icons you'll want to specify them here as well



\*

Branding it can be as simple as updating your self service icon in jamf Pro

\*

After the new self service branding takes effect the default UEX scripts look for that file and make it the icon.

\*

Also not shown here is if you've change the name or location of self service the UEX tool will update automatically as well.

## Custom icons with files

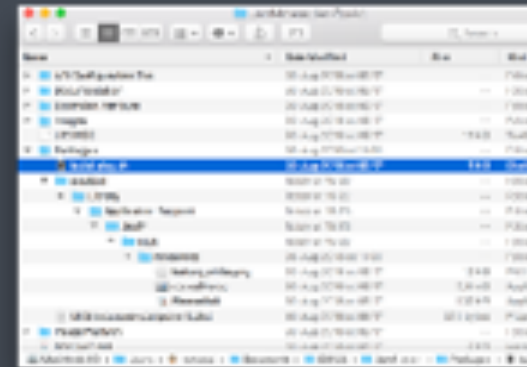
Add to the Payload  
folder

Set the owner:group to  
'root:wheel'

`chown root:wheel filename`

Set permissions to 755

`chmod 755 filename`



You can of course bundle your own icons just update the scripts I specified earlier with their path and file name.

You can bundle it into UEX resource package that automatically install if resources are missing  
It will put it in `/Library/Applications Support/Jamf/UEX/resources/`

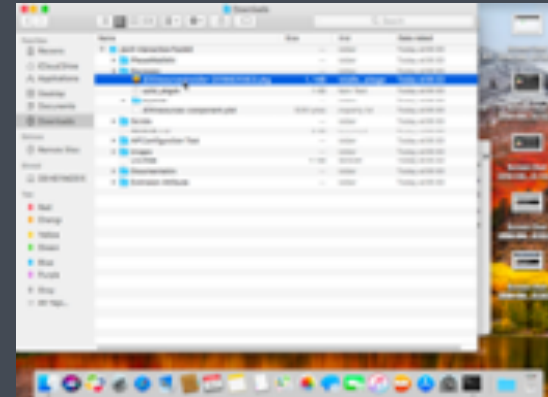


## Building UEX Resources PKG

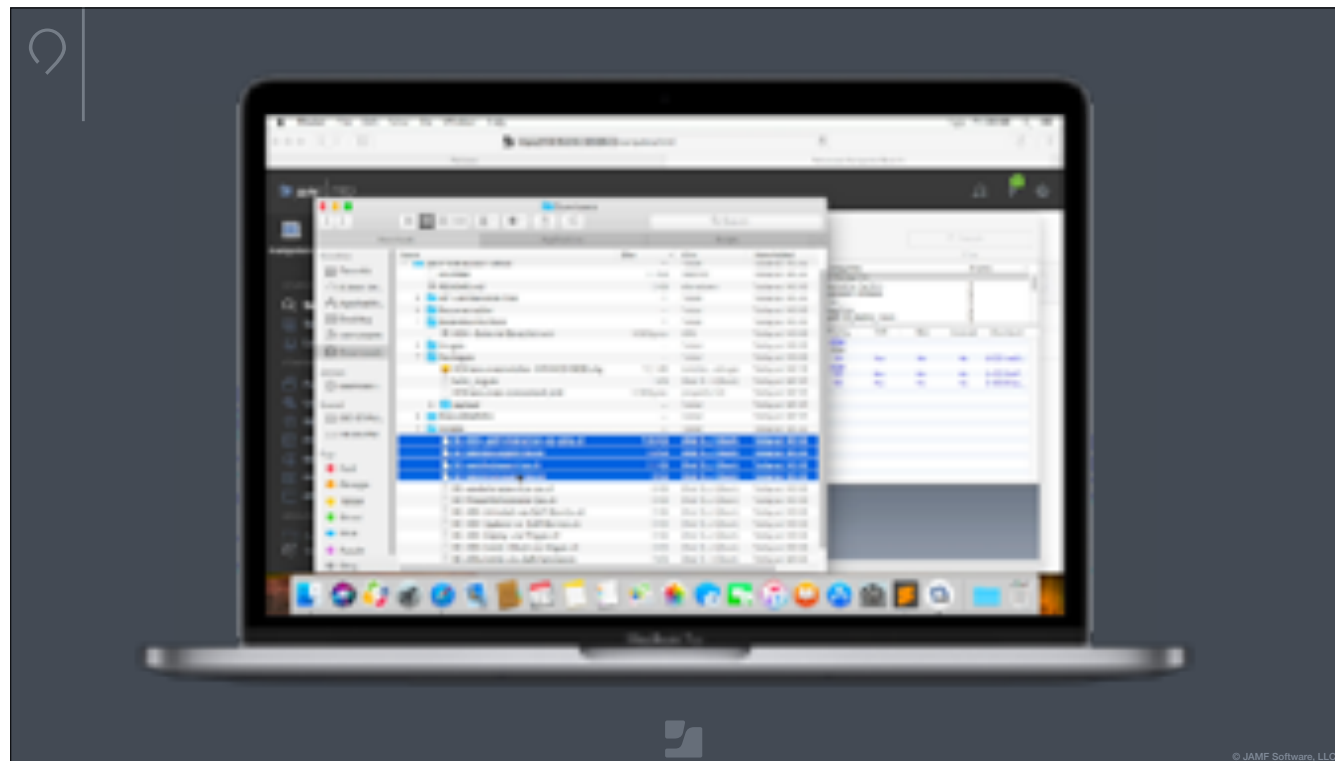
Run the build pkg script

Upload to the jamf Pro server

Don't forget to add the path to the icons into the script



To build the package it's as easy running the script I include in GitHub.



Now let's watch how this is configured on the jamf server

First update the name of your IT ORG

Upload the scripts

And upload the package

Then upload the extension attribute

Copy the name of the UEX resource package

Then you open the API configuration tool and update there

Also put in your jamf URL and credentials. Hopefully those are not your credentials there

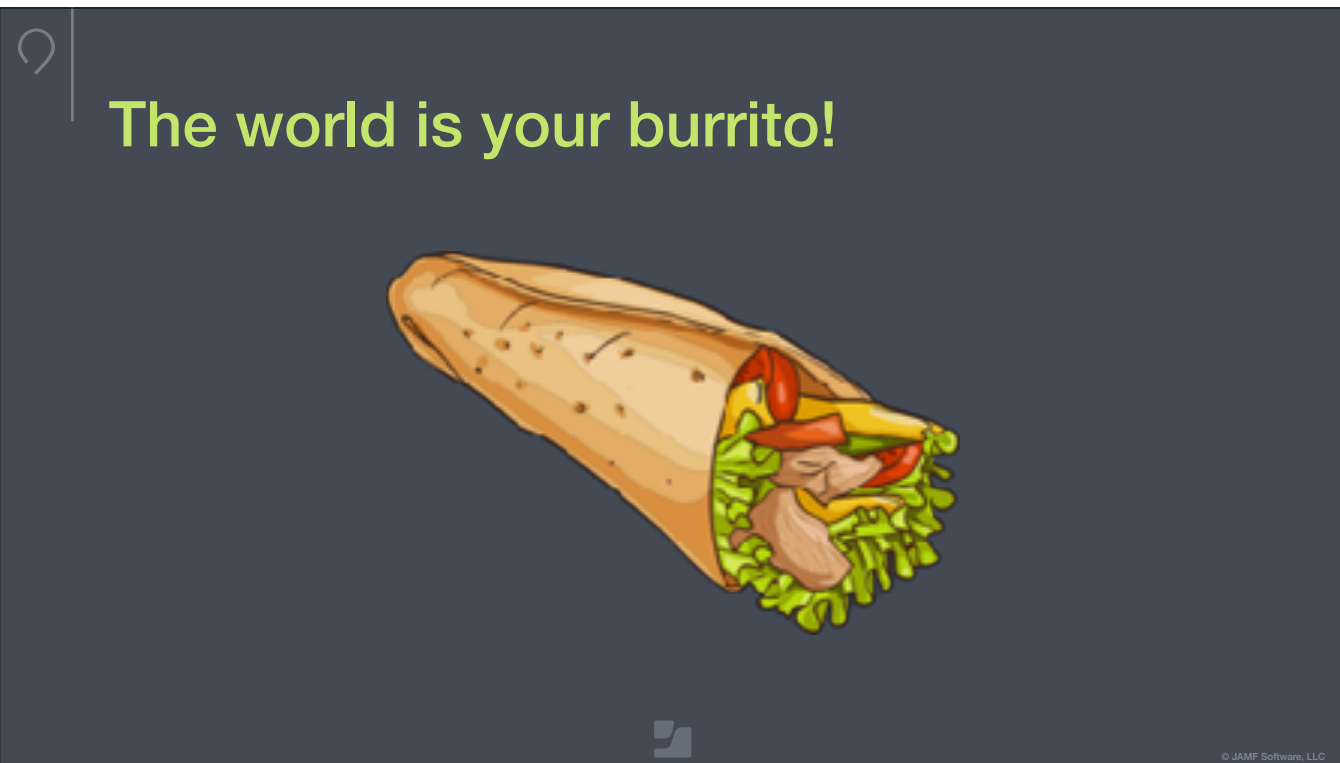
Then open up terminal and run the script.

This is not sped up at all.

Just update the scripts parameter

It creates a category to organise everything and then creates the policies and 1 smart group needed for everything to work

So that's it. That's just a slice of what you can do. So please check it out send feedback and now...



The world is your burrito.



Thank you very much for your time and attention. I'll take some questions now and do my best to answer.  
Hit me up on Slack I'm cubandave.

2

## Thank you for listening!

Give us feedback by  
completing the 2-question  
session survey in the JNUC  
2018 app.

### UP NEXT

Creating a 'Self Service First' Mentality at Target

Session time: 2:45pm



© JAMF Software, LLC