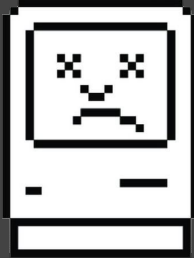


# ProfileCreator

AP Orlebeke (@apizz)

## About Me



AP Orlebeke (@apizz)

Academic Technology Integrator  
& Mac Administrator @  
The Masters School

Github <https://github.com/apizz>

Blog <https://aporlebeke.wordpress.com>

This is a little bit about me.

The Masters School is a private 5-12 day and boarding school North of NYC.

While my primary responsibility is tech integration, I'm also the primary manager of the school's Macs as well as our MDM.

# Overview

1. Overview of Current Profile-building Tools
2. ProfileCreator 101
3. Quick Review of Configuration Profiles
4. Demo
5. How to Contribute
6. Q&A

Here's an overview of what we'll be talking about.

First, I'm going to give a quick overview of the currently available profile-creation tools.

Then, I'm going to introduce you to ProfileCreator, the project, and cover it's core features.

Next, we're going to make sure we're all on the same page with regards to configuration profiles, what they are, what's in them, and why we care about them.

Then we'll actually build a few profiles with ProfileCreator, managing both Apple and third-party preferences.

Lastly, I'll talk briefly about how you can contribute and time permitting we'll build a manifest together, and have some time at the end for questions.

# Current Profile Building Tools

## GUI Tools

- Profile Manager ([macOS Server.app](#))
- [Apple Configurator 2](#)
- MDM (Jamf, FleetSmith, etc.)
- [tccprofile](#) & Jamf [PPPC Utility](#) (PPPC)

## Command Line Tools

- [mcxToProfile](#) (Custom Settings payload)
- [make-profile-pkg](#)
- [tccprofile](#) (PPPC)

If you are already familiar with profiles, then you already know that ProfileCreator joins many already well-known and established profile creating tools. Among them, Apple's Profile Manager, which is built into macOS Server but handles only macOS configuration profiles, Apple Configurator 2 for iOS & tvOS profiles, and if you have an MDM you have that profile building interface as well.

There are also several popular open-source tools, like mcxToProfile, which are great, but you have to already have a fair bit of knowledge about profiles in order to use them.

# Profile Building GUIs



Profile Manager



Apple Configurator 2



ProfileCreator



MDM

# ProfileCreator



[@erikberglund](https://twitter.com/erikberglund)

#profilecreator

[Github](https://github.com)

ProfileCreator is a macOS app (10.12 & later) developed by Erik Berglund, a member of the MacAdmin community, that allows you to build standard and customized configuration profiles for macOS, iOS, and tvOS. So out of the box, you can build profiles for all three Apple OSs from one application. This includes not only Apple preferences, but also third-party application preferences as well, which you'll see in a few moments.

I just wanted to give a shoutout to Erik. He has been very responsive to address issues, listening to feedback from the community, and adding enhancements to the app. He and I have been working together closely on this project as of late, so thank you, Erik!

I personally first stumbled on ProfileCreator in November 2018 somewhere in the MacAdmins Slack, and since then have become a very active member of the community and contributing directly to the project. When I first started to play with ProfileCreator, I quickly discovered how much less time it took to create a profile for third-party apps, like NoMAD and munki. Previously I had had to read a whole bunch of documentation, write the preferences and their values myself in a text editor, build the profile, and then ultimately test it to make sure it worked as expected.

# ProfileCreator

**“...make it easier to create configuration profiles.”**

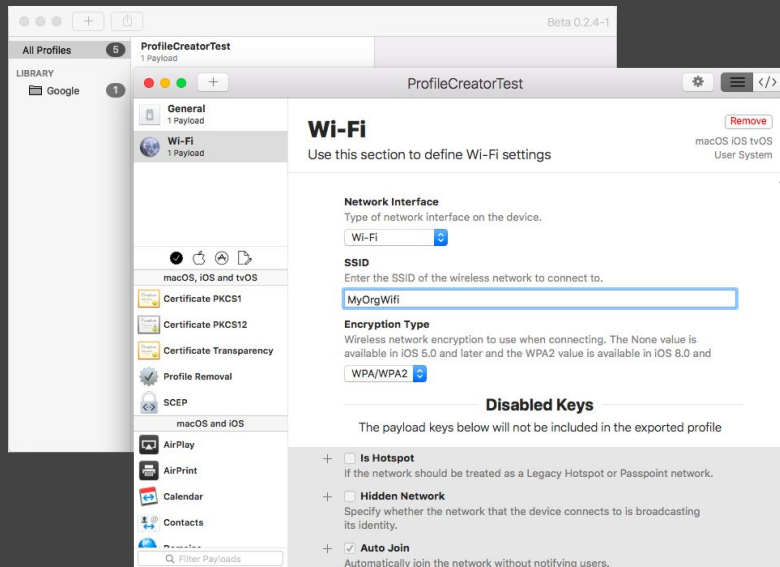
- Erik Berglund

[http://erikberglund.github.io/2018/Profile\\_Creator\\_Beta\\_1/](http://erikberglund.github.io/2018/Profile_Creator_Beta_1/)

And that's the primary goal behind ProfileCreator: to make the task of creating configuration profiles easier, for both new and veteran configuration profile builders. Ultimately we're trying to take all the things the other tools I mentioned earlier do well, the aspects that could improved, and as well as add a bunch of missing features and build them all into this one app.

While ProfileCreator is still in beta, and there are many things it does already which make it a really powerful tool.

# ProfileCreator



Here is a quick preview of the interface, which we'll look at more closely in the demo.



# ProfileCreator

## 2 Projects

- **ProfileCreator**
  - <https://github.com/erikberglund/ProfileCreator/>
  - Not currently open-source
  - [Wiki](#)
- **ProfileManifests**
  - <https://github.com/erikberglund/ProfileManifests>
  - Open-source ProfilePayloads manifest framework
  - 11 contributors
  - 77 Apple preference domains
    - Wi-Fi, Login Window, Privacy Preferences Policy Control
  - 36 third-party app domains and counting
    - Munki, NoMAD, Microsoft Office

ProfileCreator is divided into two Github projects:

1. ProfileCreator - for the app itself, which is not currently open-source. Here you can get the latest beta, access a user guide and other information from the wiki, and submit issues & feature requests for the app.
2. ProfileManifests - this project makes up the ProfilePayloads framework, which provides ProfileCreator with all the manifests it uses for each payload, like the macOS Login Window. A manifest is what contains all the settings that define each preference within a payload / preference domain, like the name of the preference and its type - boolean (true/false), string (text), etc. This is also where you can directly contribute to the project.

Thus far, there have been 11 different contributors who have created new manifests for third-party applications or updated existing manifests with new preferences, and in a few slides you'll see all the current third-party app preferences that are supported.

ProfileCreator currently supports the overwhelming majority of the native Apple preference domains - 77 in total - as well many popular third-party preferences - 36 in total -, like Munki, NoMAD, and Microsoft Office, to name a few.

# Manifests

- An XML document describing an application's preference keys that can be managed
- Used by Apple Configurator, among others
  - /Applications/Apple Configurator 2.app
  - /Contents/Frameworks/ConfigurationProfile.framework/Versions/A/Resources
- [Apple manifest format documentation](#)

A manifest is what describes the preferences that can be managed, and ProfileCreator uses this format, along with Apple Configurator and others. This is a documented standard by Apple, and which ProfileCreator in some cases builds on. So everything you see in the interface is a function of what is written within a manifest, and is where members of the community can contribute to the project. Time permitting at the end, we'll build a manifest together.

# ProfileCreator Core Features

1. User-friendly GUI w/ an XML preview
2. Manage only the specific preferences you want
3. Third-party application support
4. Helpful preference descriptions
5. Preference dependencies (x requires y; if x = ? then requires y)
6. Option to export in MCX format for Custom Settings
  - a. Can import directly into your MDM
7. Extensible framework built on Apple's [Preference Manifest format](#)
  - a. Anyone can contribute!

Core to ProfileCreator is a user-friendly graphical interface that allows you to quickly see the resulting XML code that will be produced in an exported profile. This avoids needing to export a profile and open it in a text editor to verify its contents. Additionally, you can copy the XML from this

Unlike all other graphical profile creation tools, ProfileCreator allows you to only add and manage the specific preferences you want. No longer do you need to make decisions for every single preference within a payload.

As previously mentioned, ProfileCreator supports third-party app preferences. This is a big value add because of the fact it avoids needing to spend the time reading documentation or sleuthing to discover the name of each preference you want to manage and its desired value. For new Mac admins, this makes the barrier to entry much lower.

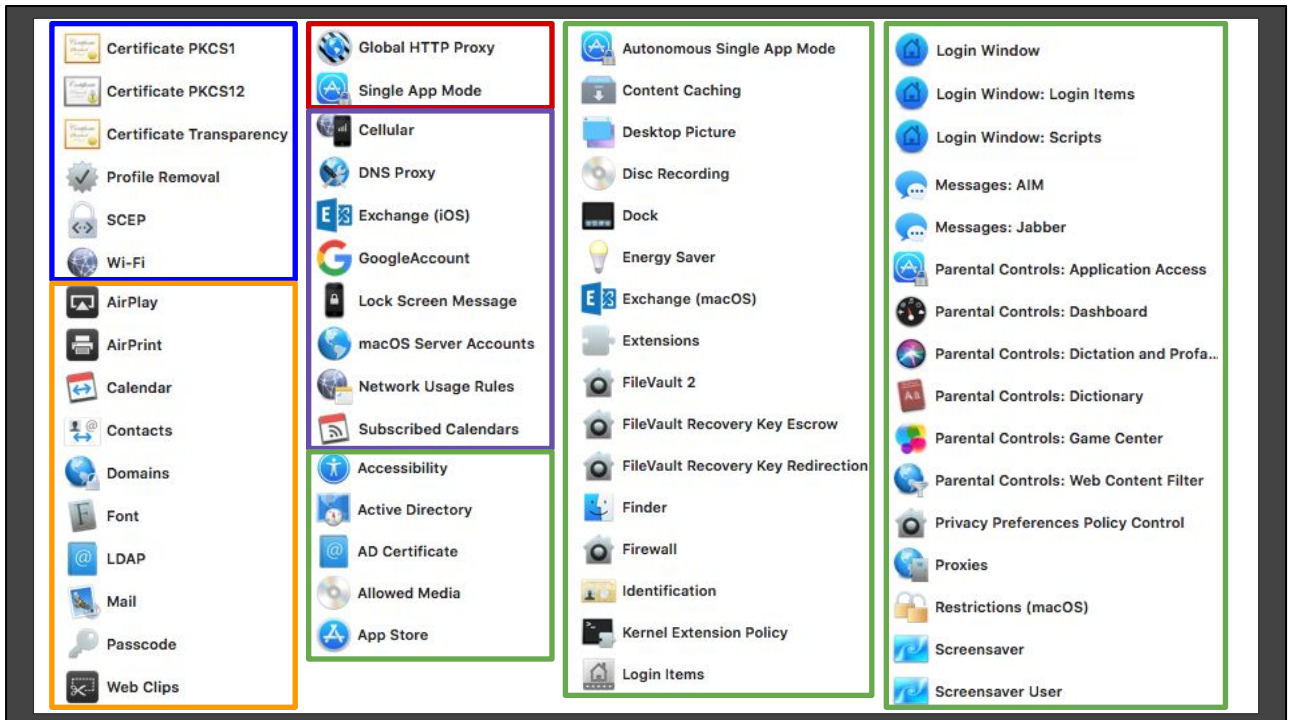
ProfileCreator also includes the option for descriptions for each preference. While most preference names make it clear what the preference controls, this isn't always the case. These descriptions help new and profile veterans have a greater understanding of what the preference deals with and other helpful details that you would otherwise have to wade through official documentation to find.

ProfileCreator also includes the ability to link preferences together. For example, if a preference requires another preference to be enabled in order to work correctly, ProfileCreator will indicate this relationship when these conditions aren't met.

Additionally, if these conditions aren't met and the profile is exported, ProfileCreator will automatically exclude these preferences from the profile.














You can choose to export created profiles in Apple's MCX style, which is the format that's used as part of the "Custom Settings" payload. Using this format when exporting a profile allows you to import profiles created with ProfileCreator directly into an MDM in this payload.

As already mentioned, ProfileCreator's ProfilePayloads framework provides the ability for anyone to contribute as new preferences are added, or additional third-party app management is desired. This is based on Apple's established Preference Manifest format, but also builds on it. This is where the real value of ProfileCreator, because rather than each of us having to figure out and maintain all these different preferences for our respective organizations, we can contribute to a single mechanism from which others can benefit. And time permitting,



Here are all the supported native Apple preference domains in ProfileCreator, continued on next slide. You'll also find this list on ProfileCreator Github wiki.

You'll notice that currently the only supported OS for the Restrictions payload is macOS. The Restrictions payload is also supported in iOS & tvOS, however these preferences haven't yet been built into the manifest. So there is still some room to grow here.

-  Setup Assistant
-  SmartCard
-  Software Update
-  Submit Diagnostic Information
-  System Migration
-  System Policy: Control
-  System Policy: Managed
-  System Preferences
-  System Preferences: Security
-  Time Machine
-  Xsan
-  AirPlay Security
-  Conference Room Display

77

## Supported Third-Party Apps & Tools

AirServer	Microsoft Error Reporting	NoMAD Login+
ANTS Framework	Microsoft Excel	NoMAD Pro
Citrix Receiver	Microsoft Office	NoMAD Shares
Crypt	Microsoft Office 365 Service	Platypus
DetectX Swift	Microsoft OneDrive	Sal
Enterprise Connect	Microsoft OneDrive Updater	Skype
Firefox	Microsoft OneNote	TextEdit
GarageBand	Microsoft Outlook	
Google Chrome	Microsoft PowerPoint	
iMovie	Microsoft Word	
jamJAR	Munki	
Logic Pro X	MunkiReport	
Microsoft AutoUpdate	NoMAD	
Microsoft AutoUpdate FBA	NoMAD Login	

**35** & counting

[Supported Payloads](#)

Here is the current list of third party apps and services that are included in ProfileCreator. So if you're using any of these applications in your environment already, ProfileCreator has you covered.

# Why Profiles?

1. Enforce desired preferences
2. Native macOS management mechanism
3. Uses macOS .plist format
4. Some preferences require MDM (PPPC)

[Apple Configuration Profile Reference](#)  
[MDM Device Settings for IT](#)

(This is more for the uninitiated) Let's take a step back for a moment and make sure we're all on the same page regarding configuration profiles. So why use profiles? Profiles allow you to enforce desired preferences. This means we're configuring the preferences the way we want them while also preventing users from changing them. This helps us ensure settings - like security and software updates - are configured per our organization's needs.

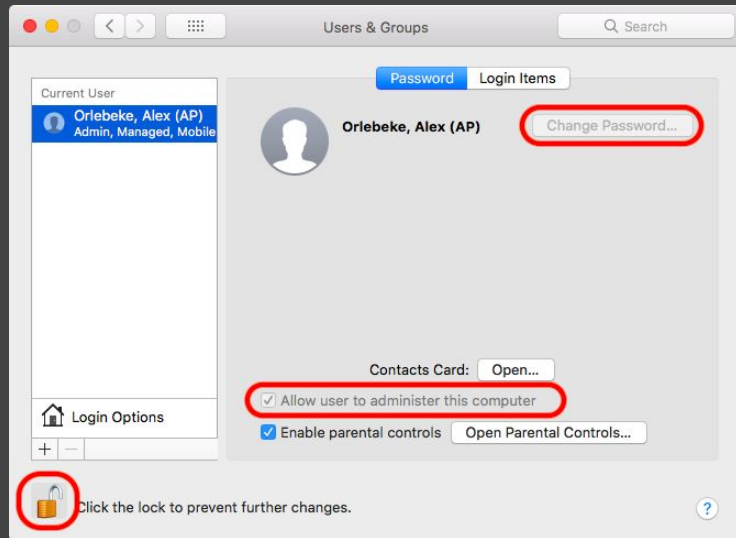
Additionally, profiles use the same property list (PLIST) format that's used throughout the system for storing user & system preferences, so if you've worked with PLISTS before (or with XML in general), this will already be familiar to you.

Lastly, with the release of macOS Mojave, Apple introduced a new security feature - PPPC (Privacy Preferences Policy Control) - which can be managed via one or more profiles. However, these profiles can *only* come from MDM. So for those folks that don't have an MDM and who use a tool like munki to handle profile installs, locally installed profiles aren't an option for this, nor are any command-line tools. As a result, it seems like profiles and MDM moving forward may be the only ways to manage new Apple preferences.

For reference, here are two links to Apple documentation on profiles.



# Why Profiles?



So, here's an example from our environment. You'll notice that despite being an admin user and having unlocked the preference pane, there are several settings that cannot be changed, as they're managed by a profile. In this case, we have a different mechanism for updating user passwords, so we don't want users to be able to do this from System Preferences, and why the "Change Password" button is greyed out.

# What's a Profile? What's in it?

- **Profile**
  - An XML file that consists of payloads that load settings and other information onto Apple devices.
  - Automates the configuration of settings, accounts, restrictions, and credentials.
- **Payload**
  - Manages specific preferences (keys & values)
    - ex. Login Window, Wi-Fi, PPPC
  - Has a number of [defined unique settings](#)
- **Key** = The preference
  - ex. Allow use of built-in camera
- **Value** = The setting
  - ex. True/False (boolean), Text (string), Number (integer), Date (date), Float (real)

(Again, for the unfamiliar) So what is a profile exactly? Some terminology so we're all on the same page:

- A profile is just an XML file, containing one or more payloads which contain the preferences to be managed.
- A payload is what manages specific preferences, like Login Window, Certificates, and Restrictions.
- The available preferences within a payload is defined by a key/value pair. The key is what defines the preference while the value is what

# Where are Profiles? (GUI)

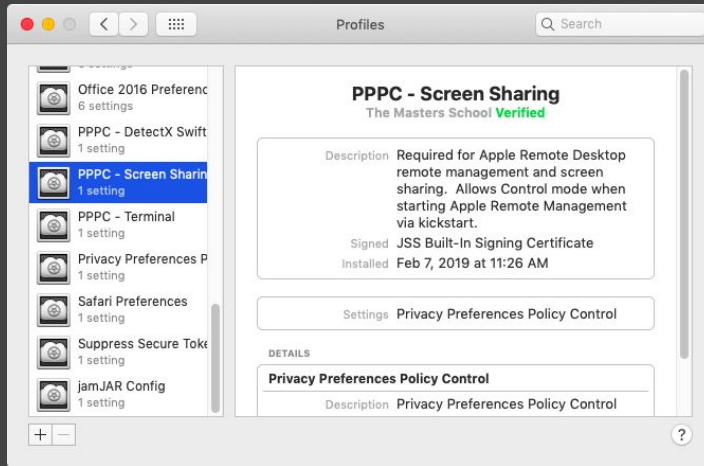
## System Preferences > Profiles



The easiest place to access all installed profiles when you're on a machine is through System Preferences.

# Where are Profiles? (GUI)

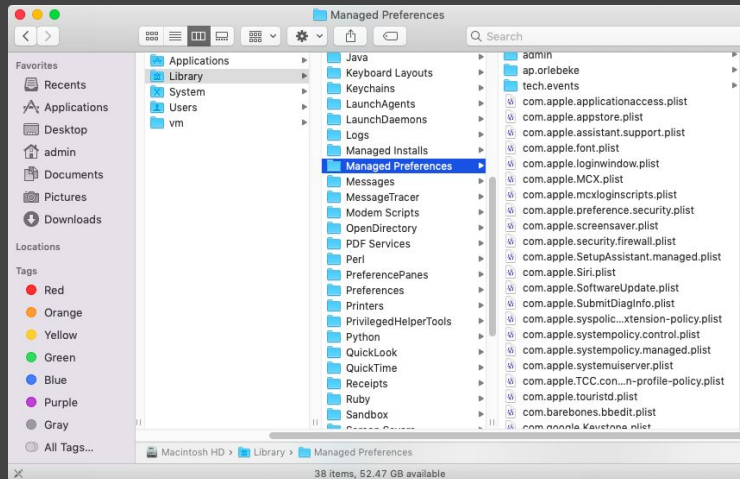
System Preferences > Profiles (includes PPC profiles in 10.14+)



With 10.14 Mojave, we can also see any PPC (Privacy Preferences Policy Control) profiles

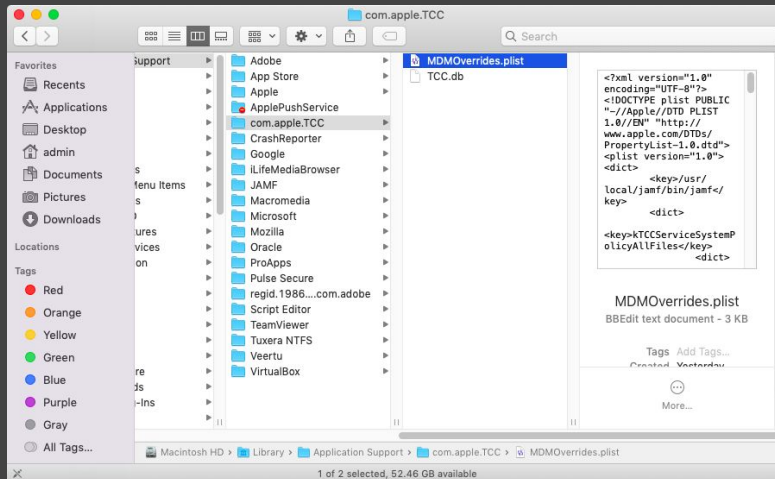
# Where are Profiles? (PLISTS)

/Library/Managed Preferences/\*.plist



# Where are Profiles? (PLISTS)

/Library/Application Support/com.apple.TCC/MDMOverrides.plist



## ProfileCreator Demo

- Build a standard macOS profile
  - Multiple payloads
- Build a standard macOS PPC profile
- Build a “Custom Settings” (MCX) profile
- Upload a Custom Settings profile into MDM

So let's take a look at ProfileCreator in action. We're going to build a couple standard macOS profiles as well as a third-party app profile. I'll also show you how you can import third-party app profiles created with ProfileCreator into an MDM.

**DEMO**



# Current Challenges

- Dependence on Apple & Vendors

Now that you've seen what ProfileCreator can do, I want to address some of the big challenges that we've faced which ProfileCreator addresses.

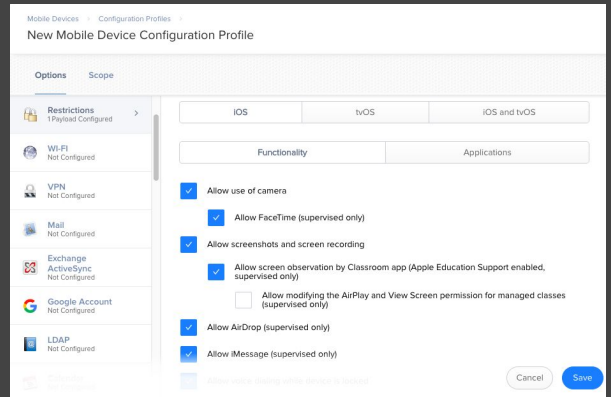
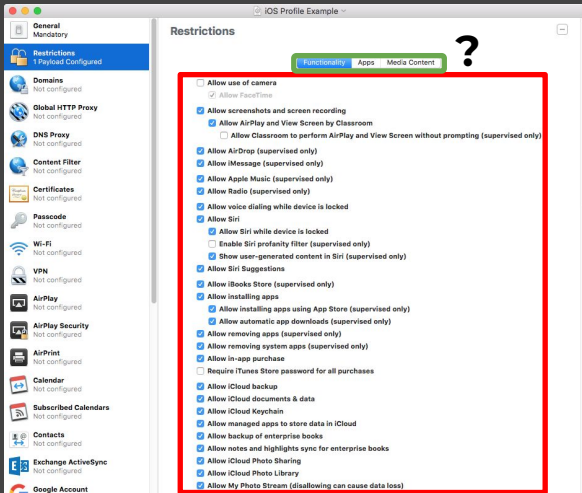
When it comes to interface, we're currently at the mercy of Apple & vendors. In the case of MDMs & Profile Manager, unless your environment makes these resources available off your LAN, you may be dependent on your org's network and/or VPN & Internet connection in order to build profiles. And as many of us know, Apple isn't always responsive to Radars and feature requests, so besides being able to manage new preferences in subsequent OS releases, the feature set in these tools are pretty limited.

Here are a few examples:

# Current Challenges - Apple & Vendors

## Apple Configurator 2

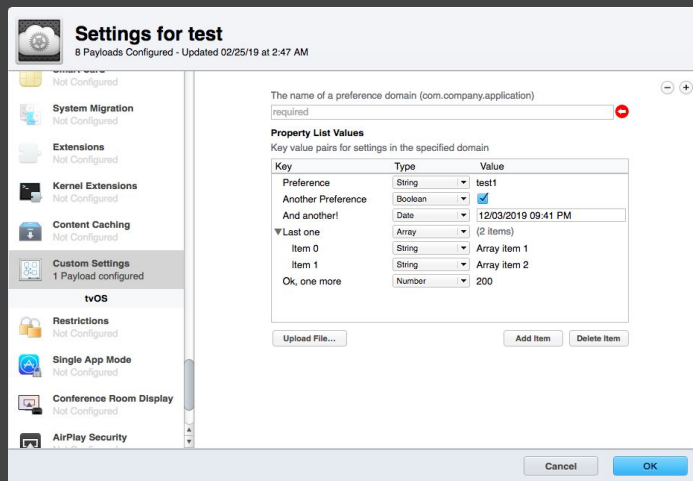
## MDM (Jamf)



As you can see in the Apple Configurator image on the left, there is some menu organization to preferences, but that it's just a long list of preferences that you have to read through in order to find what you're looking for. The same is the case in Jamf, albeit with slightly more menu structure separating different OSs.

# Current Challenges - Apple & Vendors

## Profile Manager

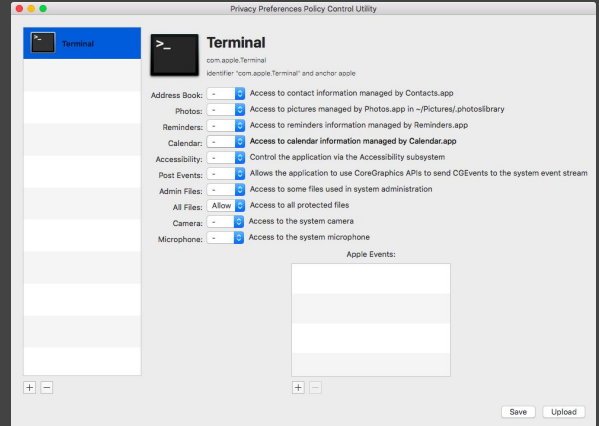
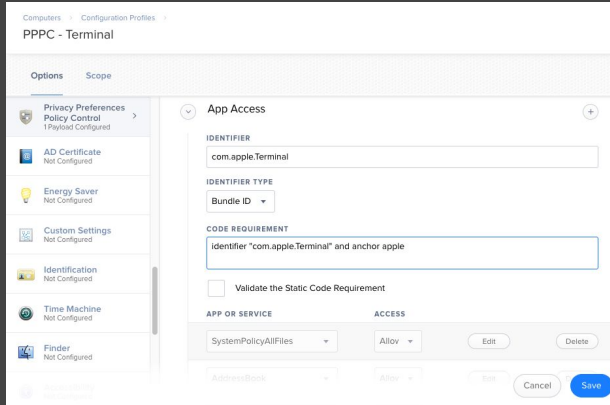


In Profile Manager here, you can see that the Custom Settings payload allows you to specify individual preferences and their types for third-party apps, but it still requires you to do a fair bit of sleuthing to determine what the preferences, preference types, and corresponding values are, which isn't ideal for new profile builders.

# Current Challenges - Apple & Vendors PPC

## MDM Web GUI (Jamf)

## Jamf PPC Utility



Additionally, with the recent introduction of PPC in Mojave, I was actually pretty disappointed with both of our MDM's solutions in the web interface and within their own dedicated utility. In the web interface on the left, you have to manually enter everything, which just isn't as efficient or user-friendly as it could be. You have to know all about the PPC structure, how to ascertain the code signature for an app ... it's just not great.

The PPC Utility is definitely better, with a drag-and-drop interface along with the ability to upload these directly into Jamf via their API, but there's still a lot of manual clicks to configure everything.

I think you'd agree that ProfileCreator handles this process better.

# Current Challenges

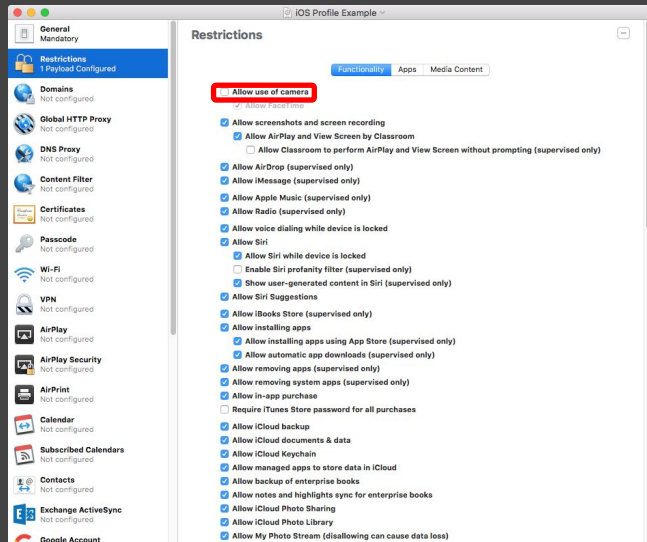
- Dependence on Apple & Vendors
- Managing 1 preference = managing many
  - No “optional” preferences, despite Apple’s own documentation
  - In certain cases, this can cause unintended collisions

When it comes to interface, we’re currently at the mercy of Apple & vendors. In the case of MDMs & Profile Manager, unless your environment makes these resources available off LAN, you’re dependent on org network and/or VPN in order to build profiles.

For the most part

# Current Challenges - Managing 1 preference

## Apple Configurator 2



So whether you are using Profile Manager, Apple Configurator 2, or an MDM, whenever you enable a payload you see a list of preferences. In the case of the Restrictions payload here, it's a very long list, and all of these preferences become managed. In this example, I just want to disable the ability for users to use the camera.

# Current Challenges - Managing 1 preference

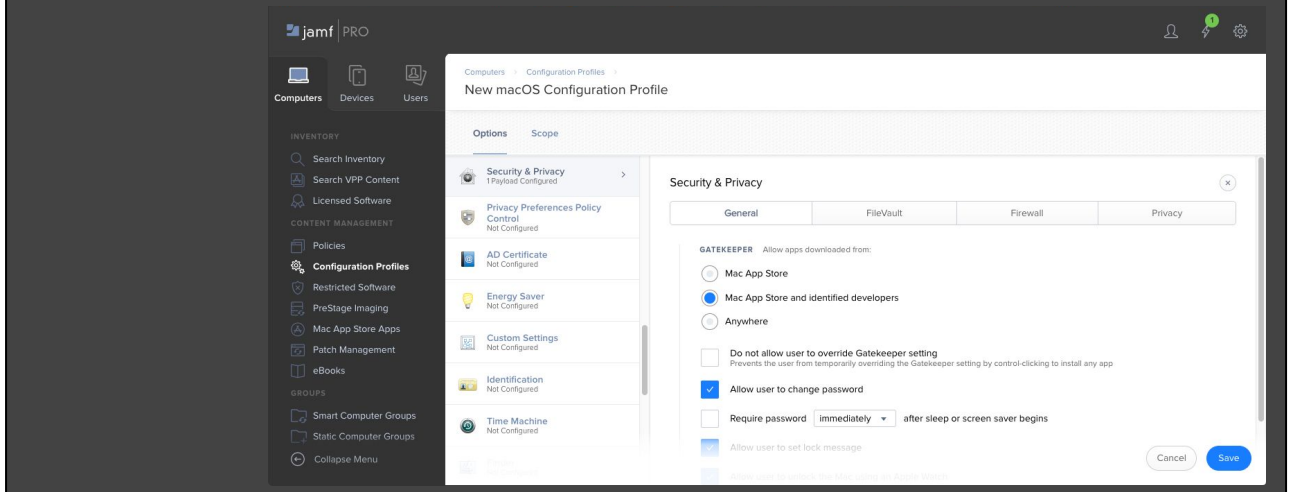
## Apple Configurator 2

```
21 <true/>
22 <key>allowAddingGameCenterFriends/<key>
23 <true/>
24 <key>allowAirPlayIncomingRequests/<key>
25 <true/>
26 <key>allowAirPrint/<key>
27 <true/>
28 <key>allowAirPrintCredentialsStorage/<key>
29 <true/>
30 <key>allowAirPrintBeaconDiscovery/<key>
31 <true/>
32 <key>allowAppCellularDataModification/<key>
33 <true/>
34 <key>allowAppInstallation/<key>
35 <true/>
36 <key>allowAppRemoval/<key>
37 <true/>
38 <key>allowAssistant/<key>
39 <true/>
40 <key>allowAssistantWhileLocked/<key>
41 <true/>
42 <key>allowAutoCorrection/<key>
43 <true/>
44 <key>allowAutomaticAppDownloads/<key>
45 <true/>
46 <key>allowBluetoothModification/<key>
47 <true/>
48 <key>allowBookstores/<key>
49 <true/>
50 <key>allowBookstoreErotica/<key>
51 <true/>
52 <key>allowCamera/<key>
53 <false/>
54 <key>allowCameraModification/<key>
55 <true/>
56 <key>allowChat/<key>
57 <true/>
58 <key>allowCloudBackup/<key>
59 <true/>
60 <key>allowCloudDocumentSync/<key>
61 <true/>
62 <key>allowCloudPhotoLibrary/<key>
63 <true/>
64 <key>allowDefinitionLookup/<key>
65 <true/>
66 <key>allowDeviceNameModification/<key>
67 <true/>
```

The resulting profile, which you can see the contents of here, include this preference, but also many others. So again, you have to make a decision regarding every possible preference within any Apple payload. Depending on your organization, you may not want to manage all of these, or be in a position to make a determination one way or another.

# Current Challenges - Unintended collisions

## Security & Privacy Payload in Jamf



I bring up this issue because it bit us in our environment early on. One of the things we do with all of our configuration profiles is we separate them such that each profile only deals with a single payload. This makes it so if we ever have to make a change to any of our profiles, we're only affecting this one payload and set of preferences. This also ends up making it easier to manage what Macs get what profiles.

However, when we were first deploying our machines that a number did not have our Login Window configured correctly. After we looked at our profiles more closely, we noticed something odd with the Security & Privacy payload ...



```
65 <string>com.apple.systempolicy.managed</string>
66 <key>PayloadUUID</key>
67 <string>EB03A3D6-379B-4D29-AE72-268C248FA538</string>
68 <key>PayloadVersion</key>
69 <integer>1</integer>
70 </dict>
71 <dict>
72 <key>PayloadDescription</key>
73 <string></string>
74 <key>PayloadDisplayName</key>
75 <string>Login Window</string>
76 <key>PayloadEnabled</key>
77 <true/>
78 <key>PayloadIdentifier</key>
79 <string>com.apple.loginwindow.EE252C94-75F1-4F4A-8B71-16CD9AB1FA39</string>
80 <key>PayloadOrganization</key>
81 <string>The Masters School</string>
82 <key>PayloadTypes</key>
83 <string>com.apple.loginwindow</string>
84 <key>PayloadUUID</key>
85 <string>EE252C94-75F1-4F4A-8B71-16CD9AB1FA39</string>
86 <key>PayloadVersion</key>
87 <integer>1</integer>
88 </dict>
89 </dict>
90 <dict>
91 <key>PayloadDescription</key>
92 <string></string>
93 <key>PayloadDisplayName</key>
94 <string>Login Window: Screen Saver Preferences</string>
95 <key>PayloadEnabled</key>
96 <true/>
97 <key>PayloadIdentifier</key>
98 <string>com.apple.screensaver.D200AE78-A4C4-4810-8FD3-F6644C0001BA</string>
99 <key>PayloadOrganization</key>
100 <string>The Masters School</string>
101 <key>PayloadTypes</key>
102 <string>com.apple.screensaver</string>
103 <key>PayloadUUID</key>
104 <string>D200AE78-A4C4-4810-8FD3-F6644C0001BA</string>
105 <key>PayloadVersion</key>
106 <integer>1</integer>
107 </dict>
108 <dict>
109 <key>AllowIdentifiedDevelopers</key>
110 <true/>
111 <key>EnableAssessment</key>
112 <true/>
113 <key>PayloadDescription</key>
<string></string>
```

Here is a portion of that profile, and as you can see, the Security & Privacy payload *also* includes the Login Window. However, no preferences are configured. And no matter what your preferences are, this payload will always be there.

The end result is that if you plan on deploying either the Login Window or Security & Privacy payloads, they must be in the same profile, at least in Jamf. The point here is that MDM vendors don't always get these things right.

## Current Challenges

- Dependence on Apple & Vendors for interface, organization, & options
- Managing 1 preference = managing many
  - No “optional” preferences, despite Apple’s own documentation
  - In certain cases, this can cause unintended collisions
- Limited to native Apple / macOS preferences and features
  - No native third-party app support

## Current Challenges

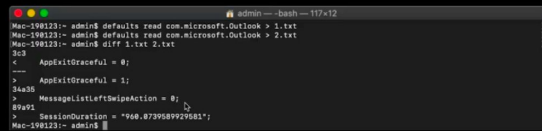
- Dependence on Apple & Vendors for interface, organization, & options
- Managing 1 preference = managing many
  - No “optional” preferences, despite Apple’s own documentation
  - In certain cases, this can cause unintended collisions
- Limited to native Apple / macOS preferences and features
  - No native third-party app support
- Sleuthing & managing third-party preferences takes time

Figuring out & managing 3rd-party preferences in most cases is time consuming. There are some very well documented 3rd party app preferences - like Microsoft Office Suite and BBEdit - but no easy way to build profiles using the preferences you want. You would have to do this manually, which is prone to errors.

# Current Challenges - Sleuthing

## Determining the Key

1. Close Outlook
2. Run `defaults read com.microsoft.Outlook > 1.txt`
3. Launch Outlook, change the value in the UI, then close
4. Run `defaults read com.microsoft.Outlook > 2.txt`
5. Run `diff 1.txt 2.txt`



```
Mac-5981231-- admin defaults read com.microsoft.Outlook > 1.txt
Mac-5981231-- admin defaults read com.microsoft.Outlook > 2.txt
Mac-5981231-- admin diff 1.txt 2.txt
diff
<
---
AppExitGraceful = 0;
---
AppExitGraceful = 1;
3a435
> MessageListLeftSwipeAction = 0;
@##V1
> SessionDuration = "946.8729589927081";
Mac-5981231-- admin
```

[How to Manage Microsoft Office 2019 for Mac \(Jamf webinar\)](#)

I took these slides from a Jamf webinar on managing preferences in Office 2019, as I thought it did a really great job of illustrating the sleuthing process. You may already be an expert sleuther, but for those who aren't ... Office for Mac is actually very well documented in terms of preferences, their format types, values, etc., but not all third-party apps can boast this.

So first we have to figure out where the preference lives. In most cases is either the root /Library/Preferences folder or user ~/Library/Preferences, but not always. With that, you can create a copy of the preferences file, make your desired preference change, and then make another copy of the preference file for comparison. While we can sometimes collect extraneous differences (AppExitGraceful & SessionDuration), you can see in the example there is a MessageListLeftSwipeAction key that has a value of 0. Because of how defaults reads preferences, the 0 could either be an integer or a boolean, so we have to go a step further and verify this preference type.

# Current Challenges - Sleuthing

## Build the Property List

1. Confirm the key and value

```
Mac-198123:~ admin$ defaults read com.microsoft.Outlook MessageListLeftSwipeAction
@
Mac-198123:~ admin$
```

2. Determine the data type

```
Mac-198123:~ admin$ defaults read-type com.microsoft.Outlook MessageListLeftSwipeAction
Type is Integer
Mac-198123:~ admin$
```

[How to Manage Microsoft Office 2019 for Mac \(Jamf webinar\)](#)

So we can use the defaults read-type argument to verify this. Just a helpful tip: because .plist files are binary files, normally if you wanted to open one in a text editor you would have to convert it to XML. However, BBEdit will open binary files just fine, converting it into the desired XML.

## Current Challenges

- Dependence on Apple & Vendors for interface, organization, & options
- Managing 1 preference = managing many
  - No “optional” preferences, despite Apple’s own documentation
  - In certain cases, this can cause unintended collisions
- Limited to native Apple / macOS preferences and features
  - No native third-party app support
- Sleuthing & managing third-party preferences takes time
- No insight in GUI as to resulting profile XML
- Sharing & contributing is limited to PLISTS & is uncentralized

# In Development

- Preference search w/in payloads
  - Make it easier to find the preference(s) you're looking for
- Different payloads GUIs for the same domain
  - Ex. com.apple.MCX - Energy Settings, Time Server, and Mobile Accounts Payloads
- Interface for going more than 4 levels deep in XML
  - Ex. Google Chrome `ManagedBookmarks`
- Decimal support for floating point numbers (real)
- More robust wiki

## On the Horizon

- Import profiles into ProfileCreator
- Merge/Split profiles
- Read preferences on disk and save as profile ([mcxToProfile](#) w/ GUI)
- Export profiles directly to an MDM using APIs
- Expandable preference key column
- Breadcrumbs & links to linked preferences

Because the preference keys don't get wider when the ProfileCreator Editor window becomes wider, in certain situations it makes it challenging to read all the text. This is most common with dictionaries that have several preference keys within



## Let's Build a Manifest! (Time Permitting)

- Enable Developer menu
- New macOS Mojave Software Update preference
  - Install app updates from the App Store
  - Different preference domain - com.apple.commerce

## Want to Get Involved?

- Join the #profilecreator Mac Admins Slack Channel
- File an issue for new / missing preferences on ProfileCreator Github
- Fork ProfileManifests & review example manifest
  - <https://github.com/erikberglund/ProfileManifests/blob/master/ManifestExamples/com.github.profilecreator.exampleApplication.plist>
- Build a manifest for a third-party app/service not yet supported

If you're looking to get involved, definitely join the #profilecreator channel in the MacAdmins Slack. There's also a number of resources in the ProfileManifests Github on the wiki for getting started and writing your own manifests. Even if you don't have time or feel comfortable writing your own manifest, please do file an issue on the ProfileCreator or ProfileManifests Github so we can eventually get these added.

**Questions?**