Block OS Updates and Installers

Whist iOS/iPadOS may only be updated using Settings, macOS can alternatively be updated using the macOS Installer. The following articles demonstrate blocking both methods.

- Fileset to block Apple Install macOS applications
- Defer Apple OS Updates

Fileset to block Apple Install macOS applications

Description

> As described, this only blocks macOS Installer Applications. Preventing users from using Software Updates can only be achieved with Defer Apple OS Updates and you should consider this because Software Update is capable or running the upgrade even with this solution in place.

Apple automatically instals the latest installer application on devices, allowing users to upgrade to the next major release of macOS. The following provides a method to prevent users from running the application, ensuring administrators have the required time to prepare the business.

> The provided Fileset includes an unaltered version of the Open Source Software Pashua, which is licensed under the 3-Clause BSD License.

Information

The attached Fileset prompts users with a message, including alternate languages. There is also allowance for control over which versions of macOS Installers are blocked. The only requirements are the following Filesets:

- mac0S - Block macOS Installer Pashua Daemon.fileset.zip

If running macOS 13 or higher, this profile is also required.

- Profile - Block Notifications

> A requirement script is included to ensure the profile is installed first, before downloading and installing the blocker.

Optionally the following Custom Field may be used to monitor the quantity of times users attempt to upgrade devices:

macOSAppInstallerBlockAttempts.customfields.zip

> The above instals launchd services. Disassociation of the Fileset will unload these services as well as remove all files.

Directions

The Fileset is currently configured to block the 'Install macOS Ventrua.app' and future versions of macOS Installer App; it would actually also stop the Beta. Version control is managed by the plist file in usr > local > etc > block_macos_updates:

```
com.filewave.blockmacosinstaller_user.plist
```

Contents of the file:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
<key>MinimumBlockedVersion</key>
<integer>18</integer>
</dict>
</plist>
```

Version of App to Block

Edit the file as required for the following:

- Key - MinimumBlockedVersion
- Value - Integer

Set to 19, which will block macOS Sonoma. This could be lowered to block earlier (or later versions when Apple release their next major release)

Example alternatives:

- 19 - Block Sonoma and above
- 18 - Block Ventura and above
- 17 - Block Monterrey and above
- 15 - Block Catalina and above
- 14 - Block Mojave and above

> ⚠ The script defines a version to block (and versions above) in the case that no plist file is found. This is set to 15, since this should never be the case and is a capture to prevent unwanted updates in this unexpected instance.

Message Localisation

When the installed service blocks the App, a message is reported to the user. Examples have been provided for English and German.

The language is determined by the first two characters from the following command:

```
$ defaults read -g AppleLanguages | awk -F "\"" '/\"/ {print $2; exit}'
en-GB
```

As such en-GB, en-US, en-AU, etc will all result in an English version.

Language template files are stored in the path:

```
/usr/local/etc/block_macos_updates/
```

English and German respectively:

- warning_en.txt
- warning_de.txt

Copy and edit the files appropriately for additional languages.

Example to add French

User has French language set:

```
$ defaults read -g AppleLanguages | awk -F "\"" '/\"/ {print $2; exit}'
fr-FR
```

Based upon this, create a copy warning file (note the suffix '_fr'):

- warning_fr.txt

Edit '*.title' and default message 'txt1.default' appropriately:

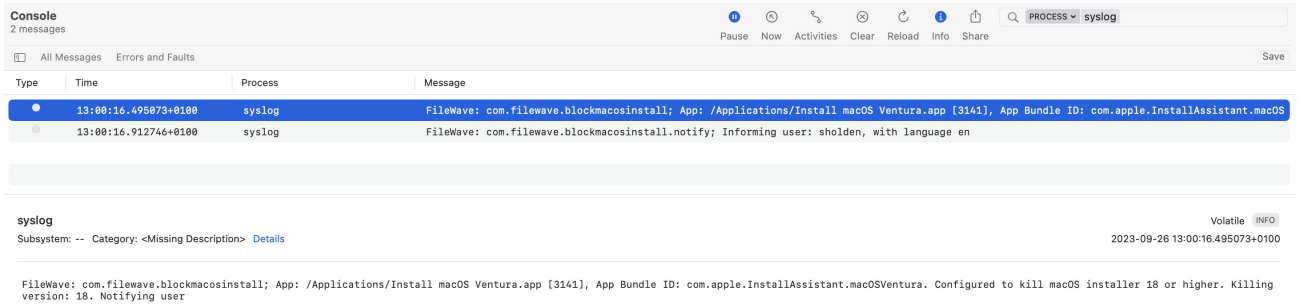
```
# Set window title
*.title = Installation bloquée

# Introductory text
txt.type = text
txt.default = macOS Installer Application
txt.height = 100
txt.width = 310
txt.x = 100
txt.y = 120

txt1.type = text
txt1.default = Cette version de macOS n'est pas prête pour l'environnement de production. Veuillez contacter le
service informatique si nécessaire.
txt1.height = 100
txt1.width = 310
txt1.x = 100
txt1.y = 50

img.type = image
img.x = 20
img.y = 70
img.maxwidth = 64
img.path = /usr/local/etc/FileWave_Icon.png
```

⚠ Text content will impact the view. Consider changing height, x and y values if the view does not appear as intended.

✔ Upload and replace the 'img.path' as your own company logo for customisation.

Logging

The launchd scripts have additional logging which will be available in Apple's Console (Debug level Info). For example:



syslog

Volatile INFO

Subsystem: -- Category: <Missing Description> Details

2023-09-26 13:00:16.495073+0100

```
FileWave: com.filewave.blockmacosinstall; App: /Applications/Install macOS Ventura.app [3141], App Bundle ID: com.apple.InstallAssistant.macOSVentura. Configured to kill macOS installer 18 or higher. Killing
version: 18. Notifying user
```

Defer Apple OS Updates

What

Apple allow users to update devices themselves, however it may be desired to delay an update, perhaps due to some incompatibility that is not yet resolved.

When/Why

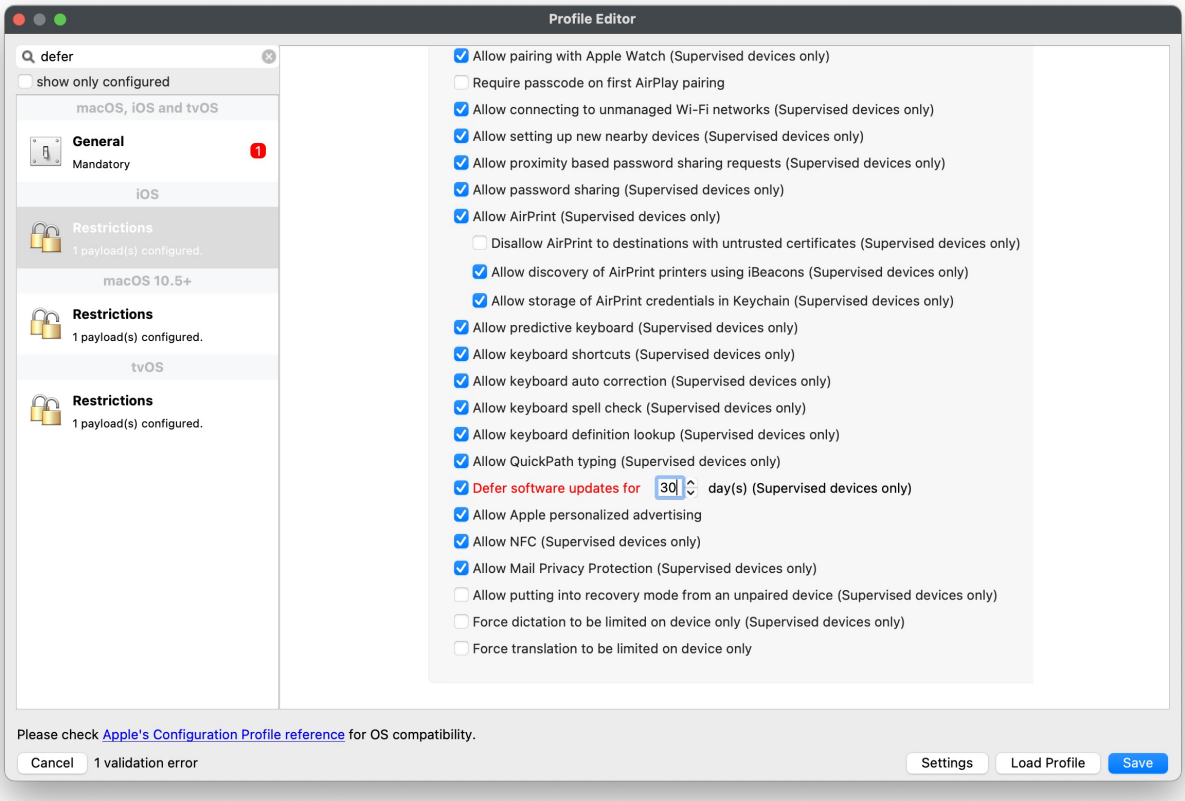
When Apple release updates, devices report these updates to the users of the device; allowing them to trigger that update as soon as they desire. Apple provide a method to 'Defer' updates. Updates deferred will no longer be visible to the user, may be deferred up to a maximum of 90 days, yet MDM may still push updates during the deferred timeframe.

> ⚠ As described, this only blocks users from using Software Updates. Preventing users from using macOS Installer Applications can be achieved with Fileset to block Apple Install macOS applications

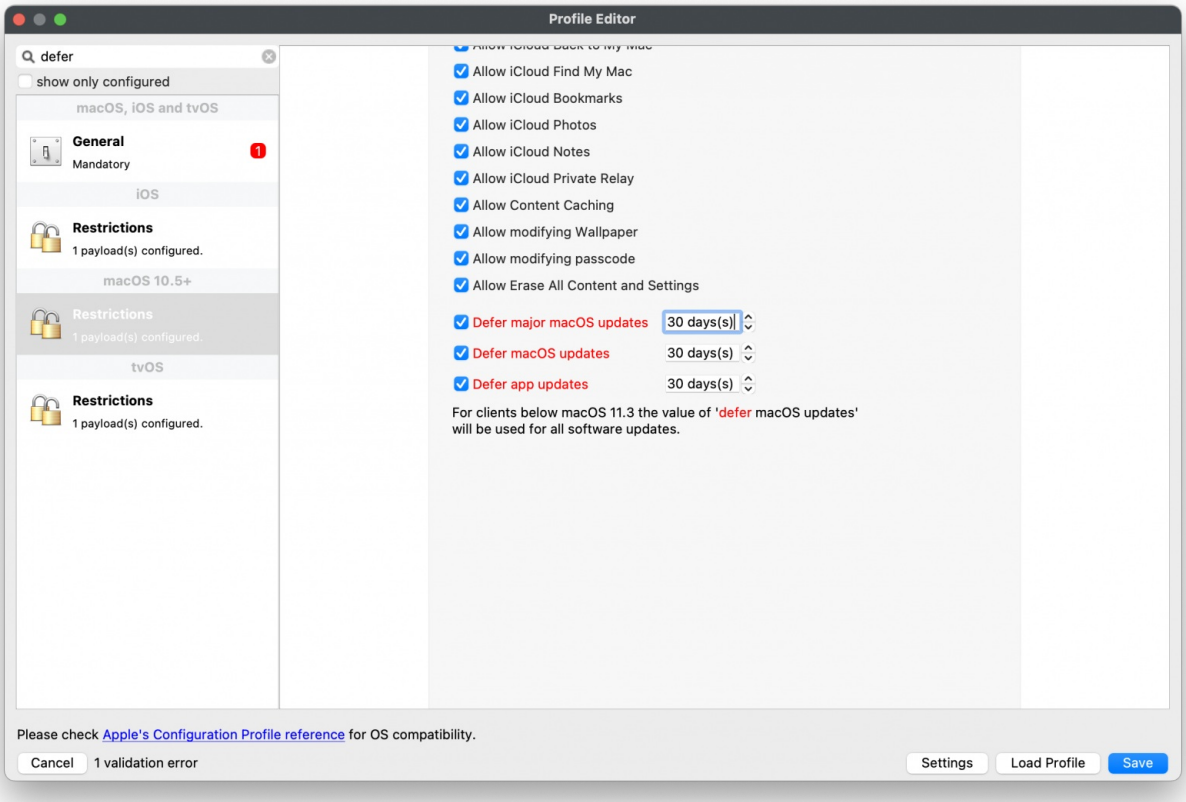
How

The method of deferral varies between macOS and mobile devices, but both are pushed using a Configuration Profile.

iOS/iPadOS



macOS



On recent macOS versions, the restriction may not apply to major updates like OS updates. Major OS updates can be hidden using the --ignore option of softwareupdate command line tool. For instance, Catalina upgrade can be made hidden by running:

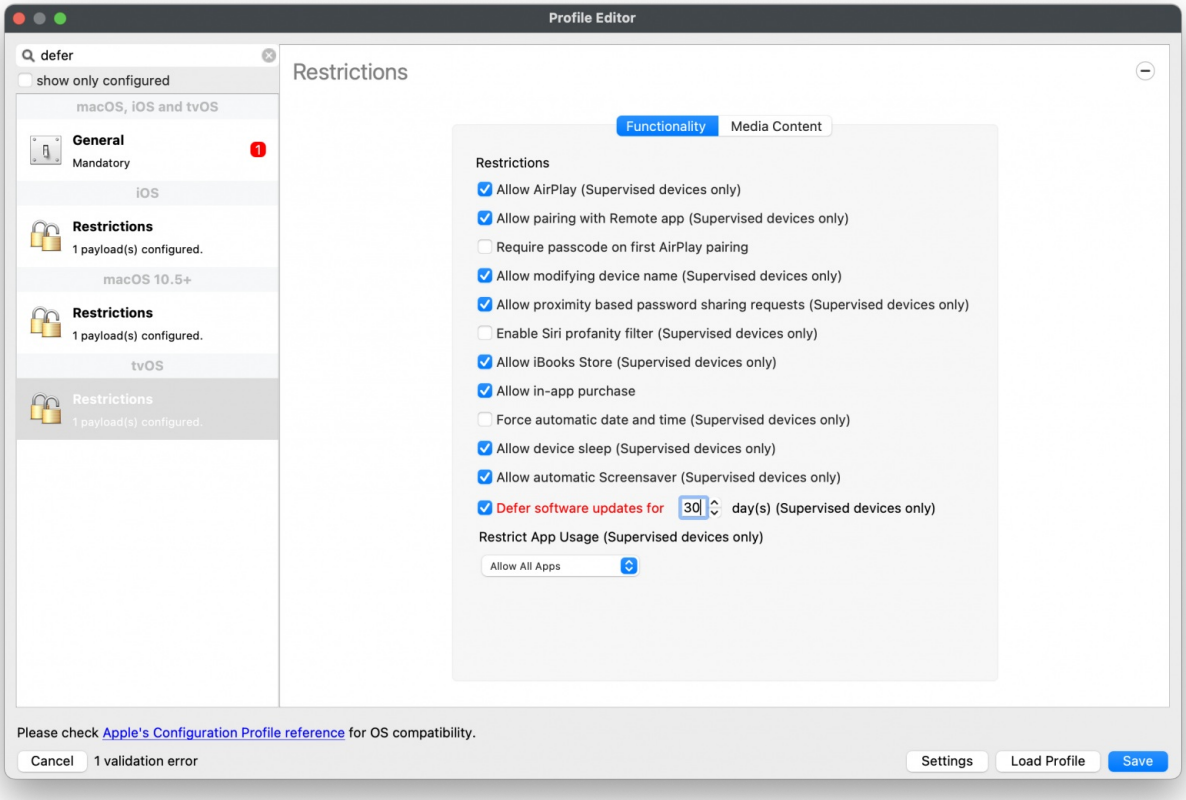
```
softwareupdate --ignore "macOS Catalina"
```

To restore hidden updates, run the following command:

```
softwareupdate --reset-ignored
```

These commands can be sent to your devices using scripts within filesets.

tvOS



Consider a recommended workflow where a defer time frame is less than maximum. This would give the option to amend the policy to a greater amount where the deadline is not met. Remember to lower the defer period, on completion of the additional testing and deployment, back to the lower chosen value.

When updates are deferred, no local tools may be used to view or instal updates released in a time frame less than the deferred period. This not only includes System Preferences, but also the 'softwareupdate' command line tool. MDM is the only method of deployment whilst an update is still in its deferred time frame.

Digging Deeper

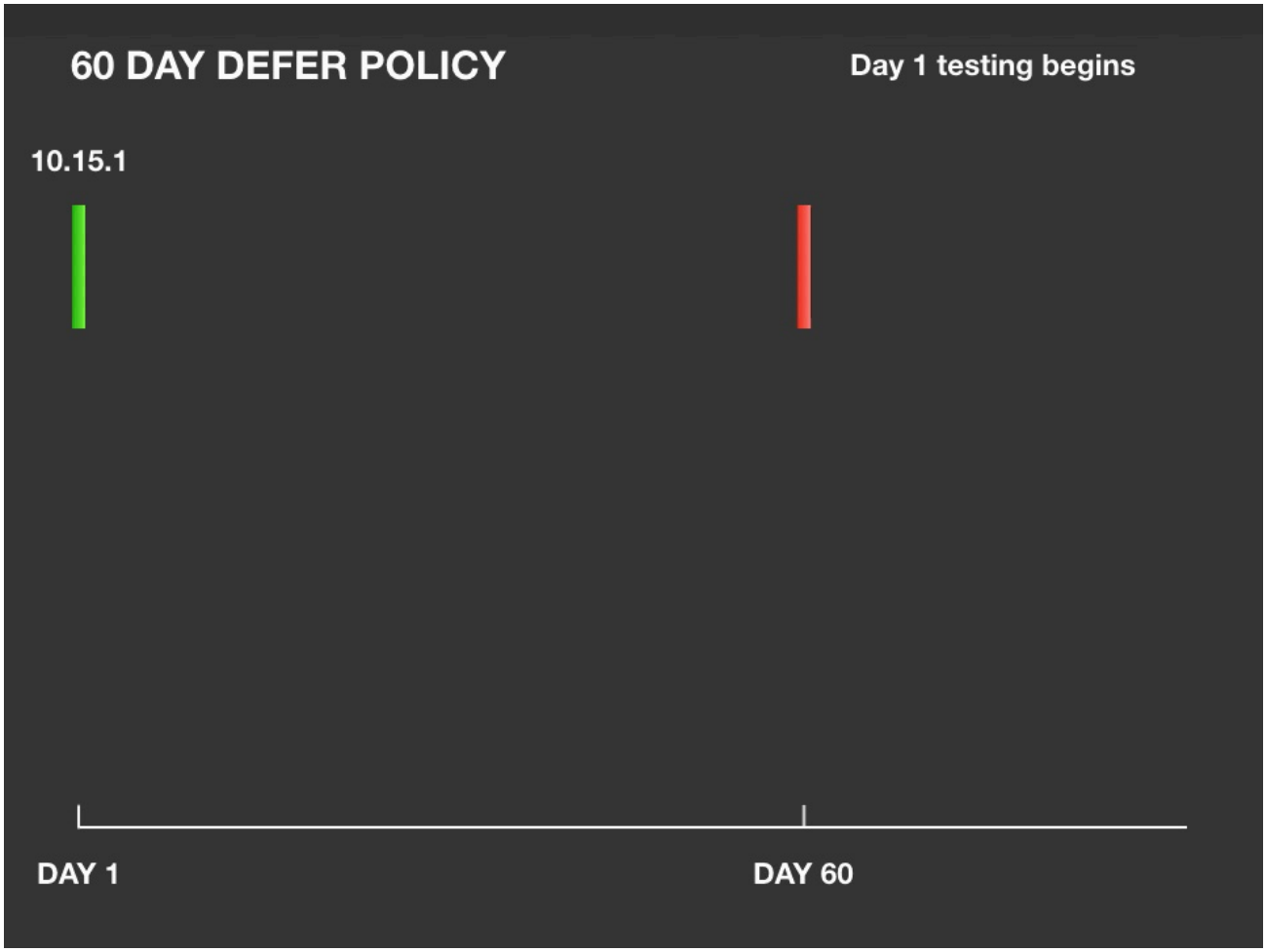
Deferring Examples

Deferment duration commences on the release date of each update. For example, with a defer period of 10 days:

- appleOS 1.1 released June 5th
- User will not be able to see appleOS 1.1 update until 15th June
- appleOS 1.2 released June 12th
- Users will not be able to see appleOS 1.2 update until 22rd June
- MDM may push out appleOS 1.1. from June 5th onwards, regardless of the deferred period
- MDM may push out appleOS 1.2. from June 12th onwards, regardless of the deferred period

Example Process

Consider a 60 day defer policy. Within 60 days after an update is released, it will not be available to the user. Once 60 days have passed, the update will show in Software Updates and the user may instal the update. It is possible that during that 60 day period an update is released to supersede the original. As long as Apple do not deprecate the former, it should still be available for deployment to devices.



In the example, 10.15.2 and 10.15.3 were released prior to 10.15.1 reaching its 60 day policy restriction. As such none of the 10.15 updates will be available to the end user. During the 60 day time frame, an MDM command may be sent to the device to instal the update; as indicated on day 50 of the example. In this instance, the command to instal 10.15.1 was sent. If not deprecated, a 10.15.1 notification should be presented to the user to instal the update.

60 days from an updates release, the policy lapses and the user is notified of the update, regardless; any devices that had not yet received the 10.15.1 deployment would then notify the user. If testing has not been completed and the 60 day policy lapses, as seen with 10.15.2, then the user will still be notified that the update exists; day 80, 60 days after day 20 when 10.15.2 was released.

It is therefore possible to be testing multiple versions of updates, prior to release to users. However, it is not possible to prevent the users from seeing those updates once the deferred period has been exceeded.