Software Updates (Apple)

The Software Updates section provides essential information and guidance on keeping your Apple products up to date with the latest operating system (OS) and third-party software updates. It covers both OS updates, including iOS, iPadOS, macOS, and tvOS, as well as updates for popular third-party applications. Stay ahead of the curve by learning how to check for and install the latest OS updates, ensuring your devices benefit from enhanced features, performance improvements, and crucial security patches. Discover tips and best practices for managing updates for third-party software, optimizing compatibility, and maximizing the stability and security of your Apple products. By regularly updating your OS and third-party software, you'll enjoy an optimized and secure experience on your Apple devices.

- Evolution of OS Updates on Apple devices (15.3+)
- Block OS Updates and Installers
    - Fileset to block Apple Install macOS applications
    - Defer Apple OS Updates

- Alternate macOS Software Update Method (Legacy)
- Apple MDM OS Software Updates
- Apple Software Lookup Service (GDMF) option for OS updates
- Apple's Rapid Security Response Software Updates
- FileWave and AutoPkg
- macOS Erase and Install Fileset (Erase Optional)
- Nudge for macOS Software Updates
- Reported Issues with macOS Software Updates
- S.U.P.E.R.M.A.N. for macOS Software Updates (macOS Script)
- Software Updates in the age of macOS MDM (Big Sur v11.0+ / iOS 15+)
- Troubleshooting dead MDM Client on macOS using launchctl kickstart Command

Evolution of OS Updates on Apple devices (15.3+)

Despite being a critical task in Endpoint Management, OS Update management is unfortunately quite a chaotic journey.

The days of merged-1.sucatalog.gz and /usr/sbin/softwareupdate.

Initially, macOS softwareupdate command could be used to manually control Software Updates. Update metadata would be made available as “sucatalog” file, one for each macOS version. This mechanism gave FileWave the ability craft our own sucatalog, allowing updates to be entirely hosted and controlled by your FileWave system.

MDM OS Update

On the mobile side, Apple introduced OS update via the MDM protocol. A couple of commands have been added to the protocol : AvailableOSUpdate command would query the device about the updates currently requested by a device, and ScheduleOSUpdate can be used to trigger the update process ; eventually, OSUpdateStatus can report information about the current upgrade progress. This mechanism has been made available on macOS as well, and made mandatory with macOS Big Sur.

The MDM version of OS Update management was supposed to simplify greatly the process, but has some downsides:

- all the control is on the device side. Sending “ScheduleOSUpdate” command is the only thing that could be done, and it has only a few options. MDM does not control when update happens, only when it can gently ask the device to update. And information why something went wrong and what to do to remediate the issue is very sparse. And many things could go wrong (network issue, low battery...)
- update information comes directly from devices ; this could be more reliable, but it also leads to confusion as Apple provides different updates for different devices (iPadOS on iPad Pro is not the same as on iPad 9) ; this confusion shows in FileWave where you can see all flavors of iPadOS 17.3.1 without knowing easily which version can be installed on which device. In addition, some updates could be installed while the device is not telling it requires them (see Test and defer software updates for Apple devices).

GDMF to the rescue

Apple introduced a new Software Update catalog, named GDMF (Global Device Management Framework); it exposes the list of currently available updates and the devices supporting them, which simplifies the process and provides FileWave all required information. Unfortunately, using GDMF update identifier is reported to be very unreliable when used with MDM ScheduleOSUpdate ommands.

And now, Declarative Device Management (DDM)

The new device management protocol, DDM, has now been extended to manage OS updates. It simplifies the process (there is no product identifier, just the version), and Apple assures it’s much more reliable than MDM (from our testing, it is). The only drawback of DDM OS update mechanism is that it requires iOS 17 and macOS 14. For devices not yet on macOS 14, you may refer to using Nudge or Superman.

To summarize

- legacy softwareupdate mechanism is unsupported and Apple strongly advises not using it since Catalina
- MDM ScheduleOSUpdate mechanism works quite reliably on iOS, but never worked reliably enough for macOS
- AvailableOSUpdate mechanism to report requested updates can lead to confusion compared to GDMF

In FileWave 15.3.0 we have;

FileWave 15.3.0 brings the first implementation of Apple’s new device management mechanism, Declarative Device Management (DDM). FileWave 15.3.0 will make use of the new Status Report for applications, providing quick and accurate Fileset Status updates for apps installed via MDM (App Store apps) on compatible iOS, iPadOS, and tvOS devices.

FileWave 15.3.0 therefore contains the foundations on top of which support for more DDM features are being built and will be provided in coming releases, such as Software Update management or Application installation via DDM.

As a conclusion, in FileWave 15.4.0, we will;

- Switch to GDMF as the only mechanism to report updates. Legacy sucatalog and AvailableOSUpdate mechanism will be

removed. This will simplify tremendously the Software Update Assistant by removing all duplicated versions.
- Switch to DDM as the only mechanism to manage updates on macOS. Managing updates with legacy softwareupdate did not work starting with Catalina and MDM mechanism is way too unreliable. This means that OS update management will be macOS Sonoma (and later) only.
- Switch to DDM for iOS 17 and later, and keep MDM for more ancient versions of iOS / iPadOS.

We strongly believe that controlling OS updates is a critical task and we are excited to see how Apple DDM support can solve many of the issues which have been reported over the years.

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

Description

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.

> As described, this only blocks macOS Installer Applications. Preventing users from using Software Updates can be achieved with Defer Apple OS Updates

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:

- macOS - 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:

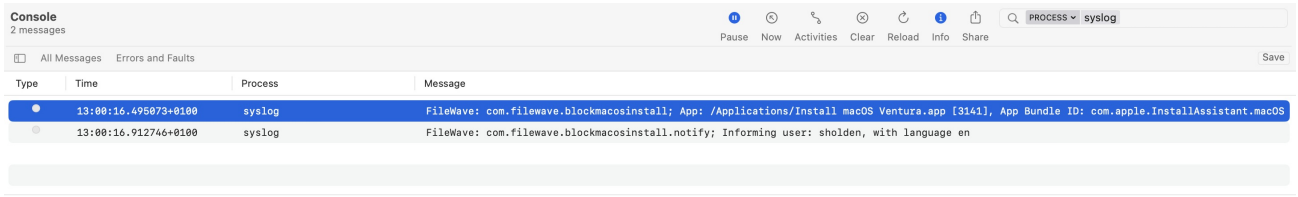
```
# Set window title
*.title = Installation bloquée

# 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

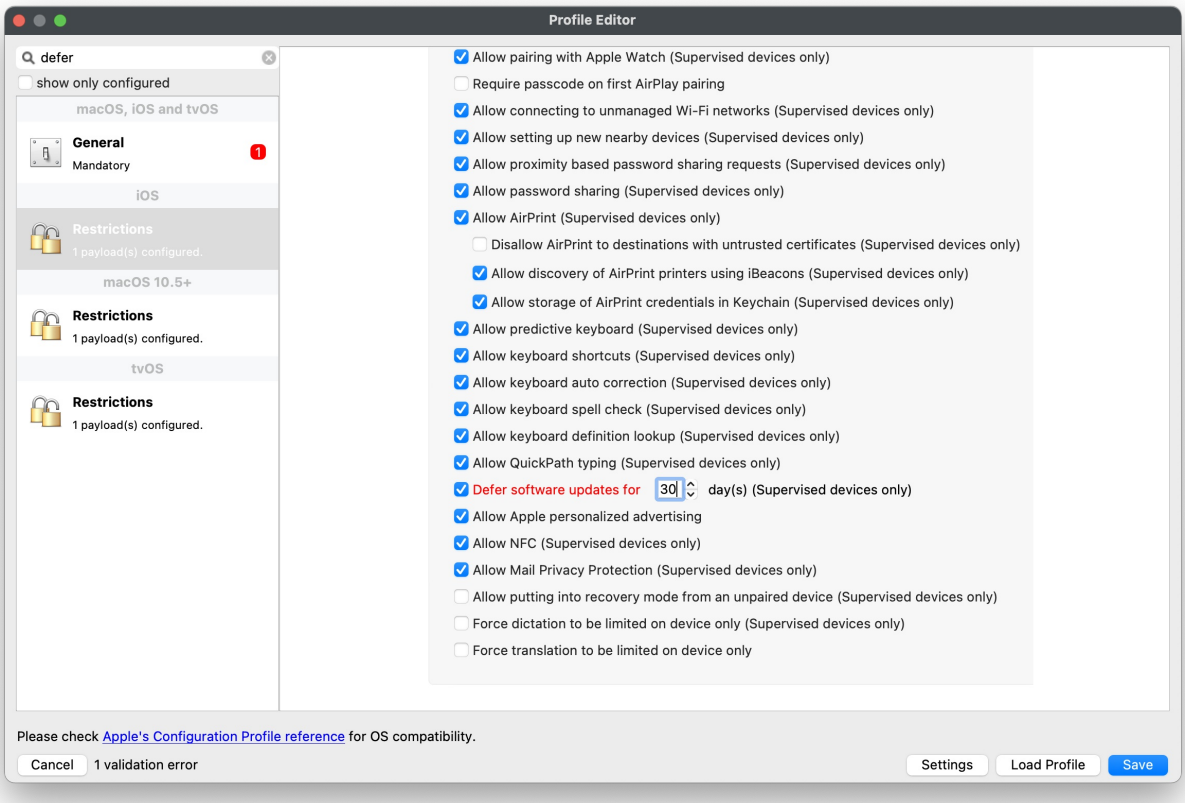
When Apple release updates, devices report these updates to the users of the device; allowing them to trigger that update as soon as they desire. Apple provide a method to 'Defer' updates. Updates deferred will no longer be visible to the user, may be deferred up to a maximum of 90 days, yet MDM may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

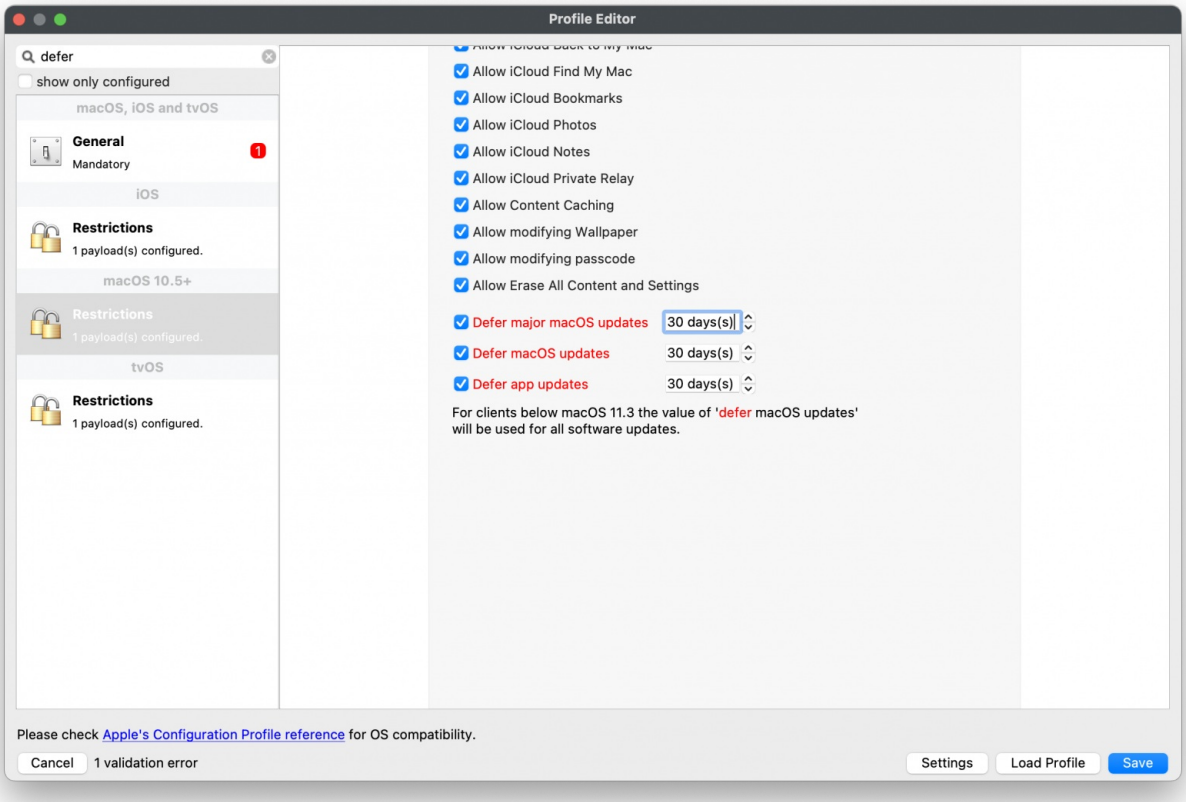
How

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:

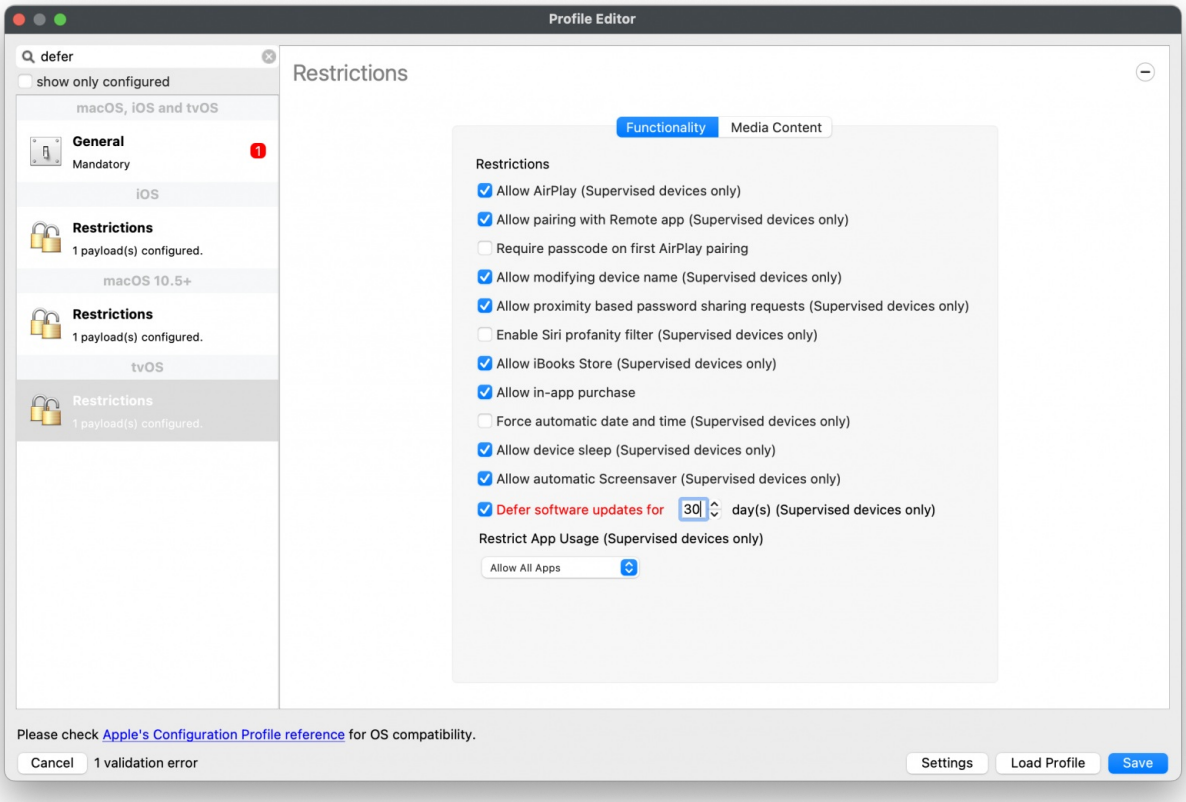
```
softwareupdate --ignore "macOS Catalina"
```

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

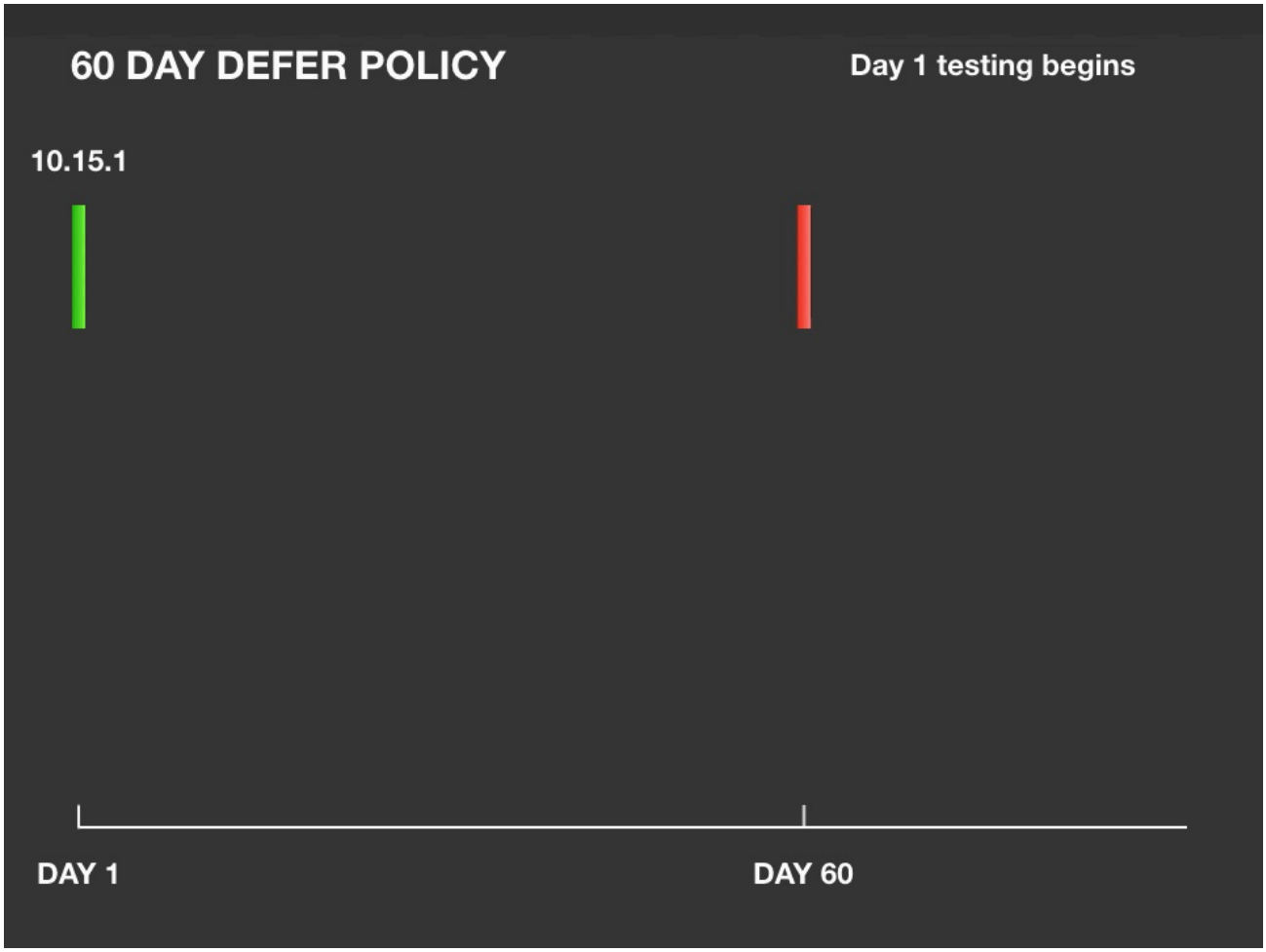
Deferring Examples

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.

Alternate macOS Software Update Method (Legacy)

Description

FileWave server automatically pulls OS Update Catalogs for computers and these are made available through the 'New Desktop Fileset'. To prevent unwanted updates being installed on incorrect machines, we would recommend using the Software Update Desktop Filesets.

As of December 13, 2017, we are aware of one particular issue (logged as engineering ticket FW-19993) that will prevent successful installation of certain macOS patch packages ("macOS 10.13.2 Update" and "iTunes" have been confirmed to be affected) using the recommended Software Update Assistant. As such, it may be necessary to create a manual fileset to deploy these updates.

Directions

The steps to follow are:

- Pull the relevant installer down from Apple's support site: https://support.apple.com
- Create an empty file set
- Place the pkg within the fileset
- Create an Activation Script to trigger the installation of the pkg
- Create a test association
- Once happy associate to only the clients that require the specific patch

Example: Upgrading a 10.13.1 computer to 10.13.2

Download the delta 'macOSUpd10.13.2.pkg' from: https://support.apple.com/kb/DL1946

(Alternatively, download the combo updater from: https://support.apple.com/kb/DL1944)

Drop the pkg into an Empty Fileset. In this example we are going to use /private/var/tmp, as this directory is naturally cleaned out on a reboot. Change the verification to 'Ignore at Verify'

Create the following script and add it as an Activation Script:

```
#!/bin/bash

# Edit the path/file to match the installer that you have downloaded
installer -pkg /private/var/tmp/macOSUpd10.13.2.pkg -target /
```

Script should be set as:

- Execute once when activated
- Wait for execution > Infinite

The fileset can now be associated to test machines. We would recommend testing on virtual machines, since once successfully updated you will no longer be able to test on that device if you wish to re-run the test.

Once satisfied it is working as expected, you may now associate to a broader range of machines.

Apple MDM OS Software Updates

What

All MDM configured devices will report and receive Software Updates via MDM. This is the same for:

- iOS
- iPadOS
- tvOS
- macOS (Either Big Sur and above or earlier versions if the client is configured for MDM Software Updates)

When/Why

Before MDM

The FileWave Server would pull the catalogue of all possible updates from Apple. This same catalogue would be delivered from the FileWave Server to each macOS client. After a Fileset containing the update was created and associated, the FileWave Client would use the Software Update process to instal associated updates, which would be sent from the Server to the device.

MDM

MDM Software Updates work mostly like Apple VPP Filesets, in as much as FileWave does not store the update. Instead, the Fileset is a reference to the update on the App Store. Only updates reported as required from devices through MDM will show in the Software Update view.

> ⚠ Only once the first requested update is reported, will the drop down option in the Software Update view show macOS updates as an option. All updates reported will persist in this view.

> ℹ The 'Requested Only' tick box will only show those updates that are believed to be currently requested by devices. Each check-in from a device should update the list of appropriate updates for that device.

> ✓ Apple catalogue updates take up storage on the server once the Fileset is created. MDM Filesets do not store the updates.

How

MDM Reporting

When necessary, FileWave server will send an APNs to Apple requesting devices check-in. On doing so, the FileWave Server will respond with MDM commands, one of which being a request for ‘AvailabeOSUpdate’. The device should reply with all possible appropriate updates. For example:

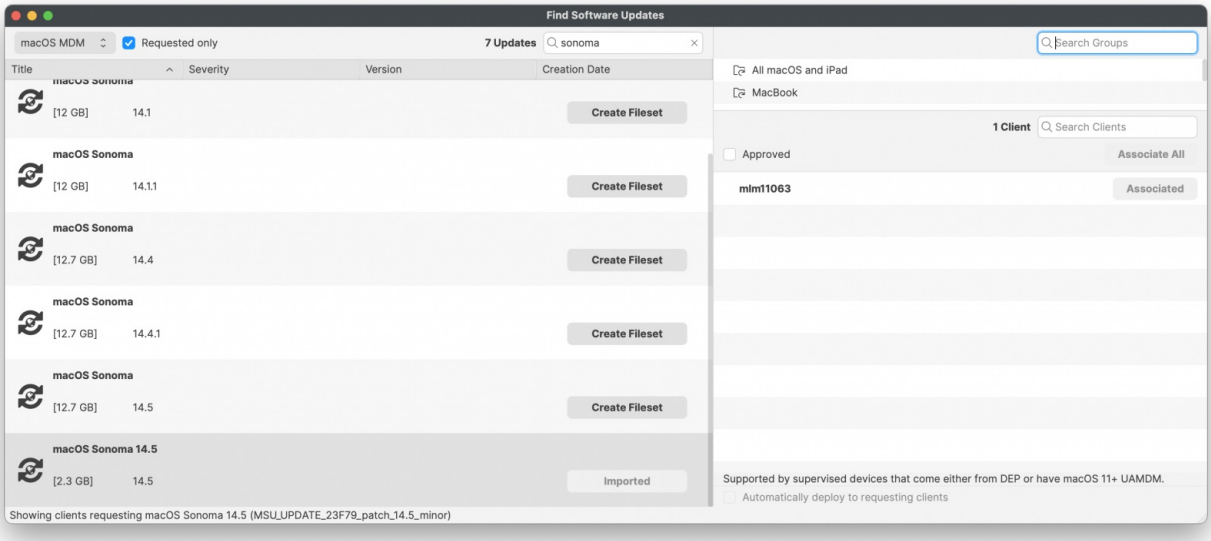
If macOS 14.5 were the latest version, a device running 14.3.1 may request all of those between:

- 14.4
- 14.4.1
- 14.5

Apple decide which prior updates will still be available. A device may be updated to any version requested, not just latest.

Software Update View

If not already visible in the Software Updates view, each reported update will be added. When an update is selected, the list of devices reporting the update as required will be displayed. From this same view, the desired update may be used to ‘Create Fileset’ and then associated.



During the transition between MDM and Catalogue Software Updates, two macOS options will be present:

- macOS - Apple Catalogue updates
- macOS MDM - Apple MDM updates

As of FileWave 15.4, only one macOS option will be present in the drop down box. Apple Catalogues will no longer be available as standard.

Fileset Association

Within the created Association is an 'Instal at' date/time. For MDM Software Updates, the command to trigger the request to instal the update will only be added to the MDM Command Queue once this date/time has been reached. This is not an exact defined time, since APNs check-in request and device response may not be until a day/time after this. Additionally, other factors can impact the device's reaction to any such request, e.g batter power.

MDM Software Updates and Rapid Security Responses are impacted by battery percentage of the device. If not on charge, there is a minimum percentage required for the update to commence. Please see the below chart.

Mac Notebook Type	ASU Battery Requirement	RSR Battery Requirement
Mac with Apple silicon	20% • Priority key set as High	10%
Mac with Apple silicon	50% • Priority key not set	
Intel Based Mac	50%	20%

DDM

FileWave introduced additional software update features with version 15.4, in particular, Force Reboot through Apple's new DDM protocol. If the user has not actioned the update prior to this time, the device should then force this installation.

'Instal at' date/time, if not set, defaults to the time the association is created. The 'Instal at' date/time in FileWave 15.4 is now a Force Reboot time. Alter this time if this is undesirable.

The FileWave option to 'Automatically Instal on new devices' does not show the association. It is therefore not possible to alter this date/time.

MDM Process

Since FileWave does not deliver the update to devices, but a reference to the update on the App Store, devices must first fetch the update. Therefore, the installation process of an update is twofold, one to download the update and another to instal the update.

Since updates can be very large, there can naturally be some delay between the initial association and the actual installation.

> If the device were upgraded between the association being created and the device receiving the command to update, if the update is no longer appropriate, the device will silently ignore it.

Apple Software Lookup Service (GDMF) option for OS updates

What

Before FileWave 14.7+ the MDM protocol only displayed updates reported by devices. Apple's GDMF allows FileWave to display all currently available updates. The former only allowed for installing the latest version reported by the device, while GDMF provides provision for intermediate updates that are still available from Apple.

> FileWave now reports multiple updates for devices and all device types will also report iOSUpdate as a named update, where previously they would only report a dedicated updated, for example: iPadOSUpdate

When/Why

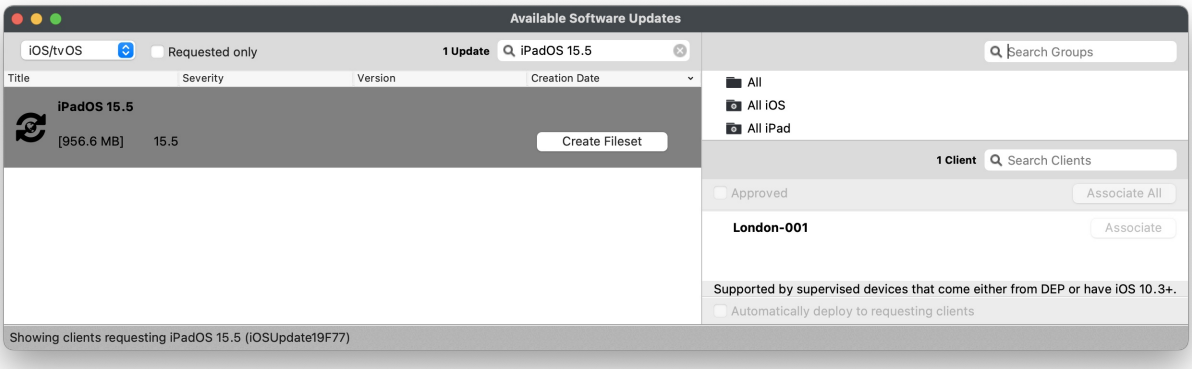
This feature allows upgrading an iPad running iPadOS 15.0.2 to iPadOS 15.1 while the most recent update might be 15.3, for example. This functionality is supported on iOS, tvOS, and iPadOS.

Device Reports

Historically, FileWave would receive the latest reported update from a device. For example, any compatible iPads running an iPadOS version of 15.0.2 or below would report the following where 15.3.1 is the latest available version:

```
        <key>AvailableOSUpdates</key>
        <array>
                <dict>
                        <key>AllowsInstallLater</key>
                        <false/>
                        <key>Build</key>
                        <string>19D52</string>
                        <key>DownloadSize</key>
                        <integer>667505362</integer>
                        <key>HumanReadableName</key>
                        <string>iPadOS 15.3.1</string>
                        <key>InstallSize</key>
                        <integer>673185792</integer>
                        <key>IsCritical</key>
                        <false/>
                        <key>ProductKey</key>
                        <string>iOSUpdate19D52</string>
                        <key>ProductName</key>
                        <string>iOS</string>
                        <key>RestartRequired</key>
                        <true/>
                        <key>Version</key>
                        <string>15.3.1</string>
                </dict>
```

Note the device reports not just the update version, but the update size also. FileWave shows the Human Readable Name reported for this update and the installer's size. The Product Key is shown in the footer of the window where an update is selected.

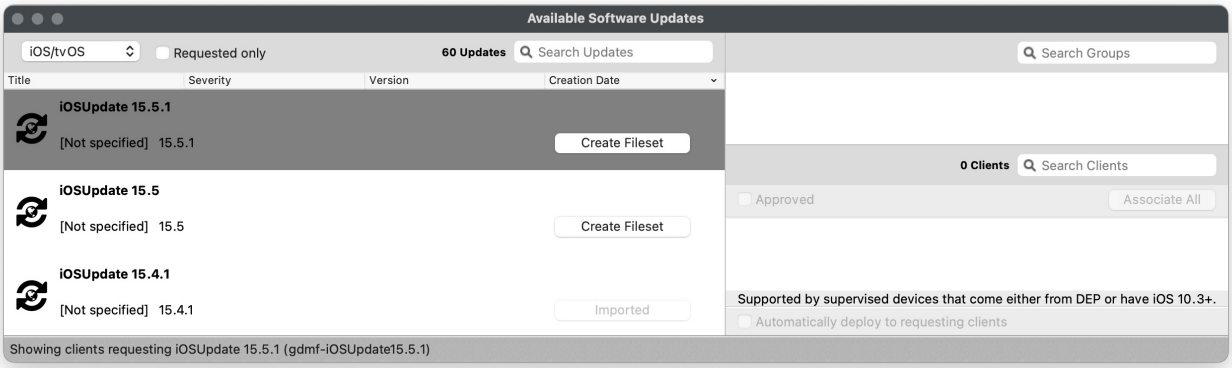


As such, only updates reported by devices were visible and only subsequently could the Fileset association be created and associated.

GDMF

GDMF updates are pulled from a URL (akin to the mechanism macOS used prior to MDM updates). All mobile updates currently available from Apple are listed under one generic name: 'iOSUpdate', followed by the version number. Devices need not report required updates in advance and associations can be made prior to the device's next check-in, benefitting from the update being already available and associated.

The catalogue of data provided by Apple does not include the file size of the update, but FileWave populates the footer of the update window to indicate the update is a GDMF update when selected.



How

FileWave server will regularly check Apple GDMF service to pull the list of available updates for each device. Make sure to read and follow Apple documentation related to network requirements. GDMF service provides the following information:

- Product version
- Posting date
- Expiration date
- List of supported devices

For instance, the following shows extract for iOSUpdate 15.3:

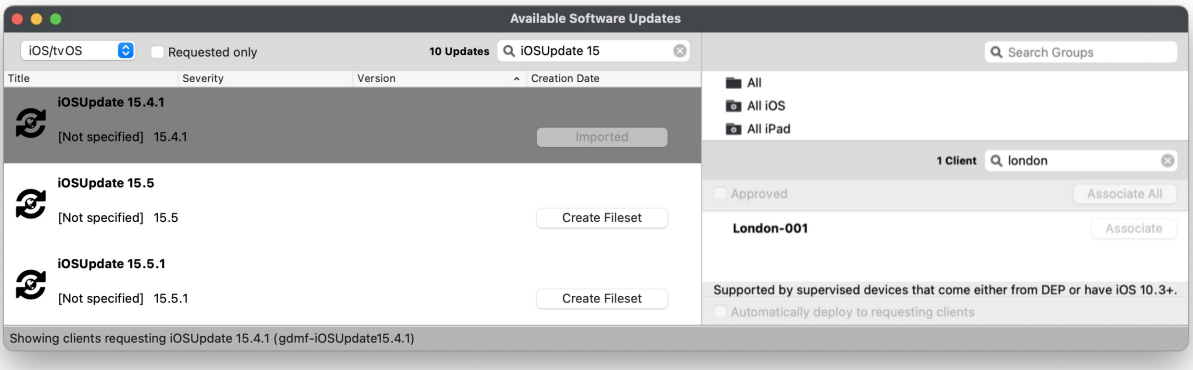

```
{
    "ProductVersion": "15.3",
    "PostingDate": "2022-01-26",
    "ExpirationDate": "2022-05-11",
    "SupportedDevices": [
      "iPad11,1",
      "iPad11,2",
      "iPad11,3",
      "iPad11,4",
      "iPad11,6",
      "iPad11,7",
      "iPad12,1",
      "iPad12,2",
```

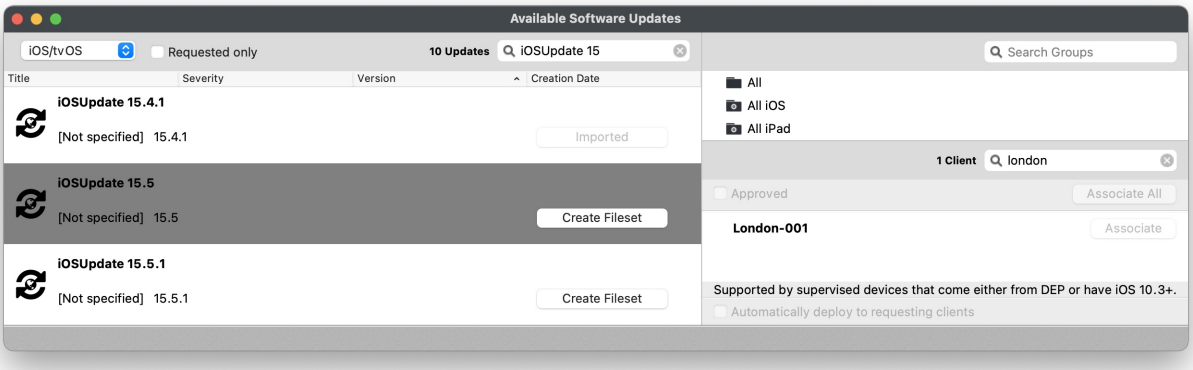
```
            "iPad13,1",
            "iPad13,10",
            "iPad13,11",
            "iPad13,2",
            "iPad13,4",
            "iPad13,5",
```

FileWave will cross reference all supported device types for each Product Version and show these as requested by the device, even though the device may not actually have personally requested them. This allows pre-association.

> FileWave will update the Software Update view on every synchronisation with Apple. Only updates currently available should be reported. If Apple deprecate an update, this update should no longer show in the Software Update view. Cross reference this view with currently Created/Associated Filesets to ensure the updates are still current.

GDMF advantageously proposes all possible updates for each device, not just the last one, as can be seen in the below images.
Device 'London-001' shows both iOSUpdate 15.5 and 15.4.1 as appropriate:





You can now follow the usual process : create Fileset, approve the update, associate the update for any currently available intermediate version.

Considerations

Reported Updates

Note, not all updates are necessarily appropriate for all device types. For example, the current version of 15.5.1 at the time of writing was only appropriate for these device types. Of course, Apple may change this over time, but FileWave should list appropriate devices (as listed by Apple) for any given update.

```
            {
                "ExpirationDate": "2022-08-23",
                "PostingDate": "2022-05-25",
                "ProductVersion": "15.5.1",
                "SupportedDevices": [
                    "AppleTV11,1",
                    "AppleTV5,3",
                    "AppleTV6,2",
                    "AudioAccessory1,1",
```

```
                "AudioAccessory1,2",
                "AudioAccessory5,1"
            ]
        },
```

Create Fileset

FileWave now reports both GDMF and Client Reported updates. Consequence of creation:

Device Reported Updates

Creating a Fileset, based upon those the devices report as requested, will only attempt to push this update whilst the device still requests this update. The consequence for any device not yet updated, when the next version is released, is this update Fileset will no longer be successful, since the device will no longer request this version.

GDMF Updates

Creating a Fileset based upon a GDMF reported Fileset allows any displayed version to be pushed, regardless of the latest available version. This will only be true whilst Apple maintain this update online. Once Apple deprecate this update the matching Fileset will no longer be able to update any devices not yet upgraded. Below is the list of possible iOS 15 updates available at the time of writing. Notice that version 15 updates below 15.3 are no longer available, whilst a version of 14 is still available.

Example of Currently Available Updates

```
            "ProductVersion": "15.5.1",
            "ProductVersion": "15.5",
            "ProductVersion": "15.4.1",
            "ProductVersion": "15.4",
            "ProductVersion": "15.3.1",
            "ProductVersion": "15.3",
            "ProductVersion": "14.8.1"
```

Avoid associating multiple appropriate updates. It is unclear in this instance what may occur and is possible one update will cancel the first ending in a situation where no update is actioned. If automatically deploying to devices is chosen, disable this before assigning a new association.

Consider using one method only, either the original reported version by device or GDMF. GDMF provides the greatest scope of control though and would probably be the better choice going forward in most environments.

Related Content

- MDM Software Update Changes

Apple's Rapid Security Response Software Updates

What

Apple is known for its high standards of security and privacy for its users. However, no system is perfect and vulnerabilities can still be found and exploited by malicious actors. That's why Apple has developed a Rapid Security Response (RSR) process that allows it to quickly identify, fix and deploy security updates to its devices.

What is RSR?

Rapid Security Response (RSR) is a method for deploying security fixes to users more frequently. RSR is a process that Apple follows when it becomes aware of a security issue that affects its products or platforms. It involves four main steps:

- Investigation: Apple's security team analyzes the issue and determines its severity, impact and scope.
- Mitigation: Apple's engineers work on developing a patch or workaround to address the issue and prevent further exploitation.
- Testing: Apple's quality assurance team tests the patch or workaround to ensure it works as intended and does not introduce new problems.
- Deployment: Apple's release team distributes the patch or workaround to its users via software updates, security bulletins or other channels.

When/Why

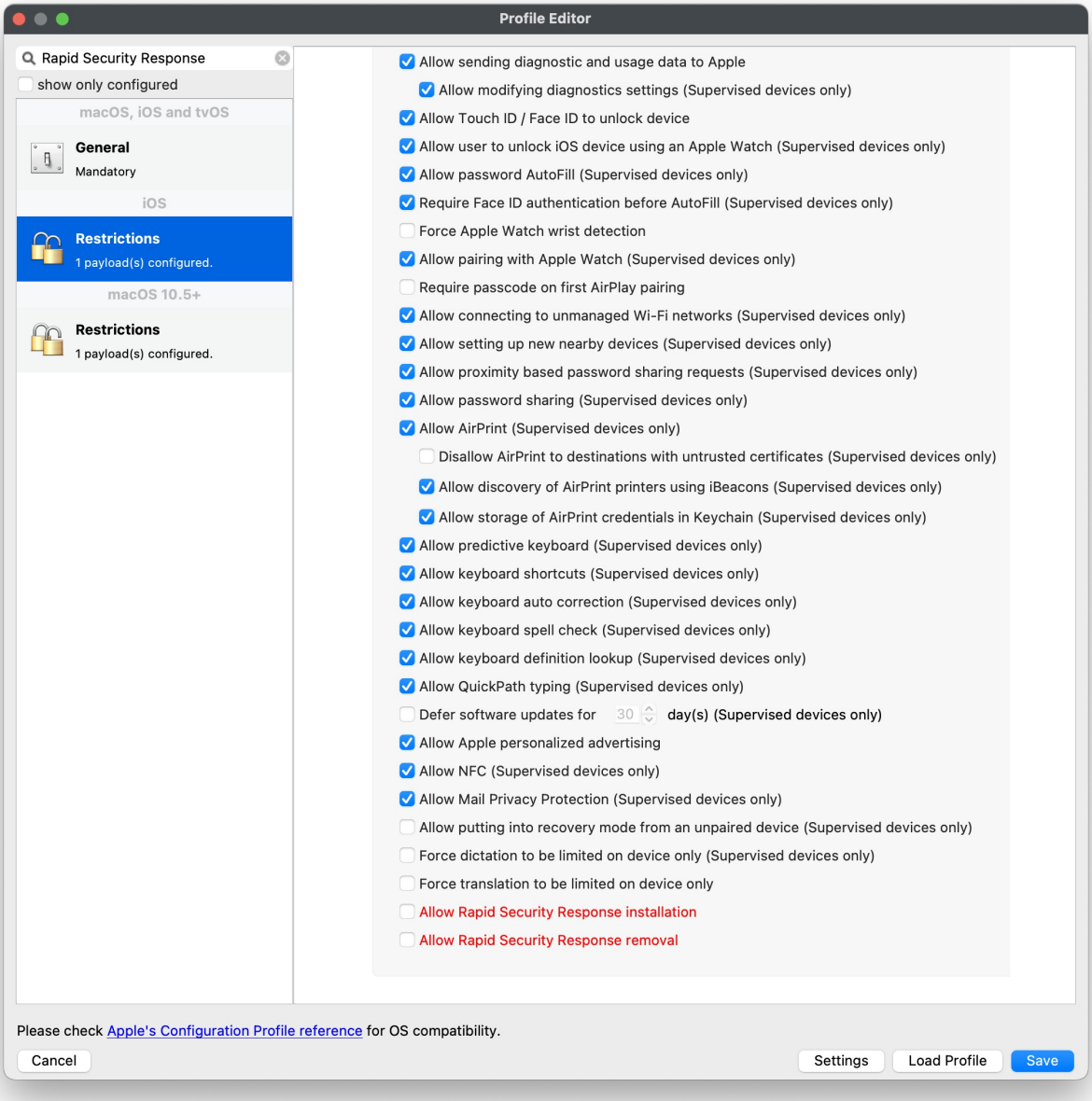
RSR is important because it helps Apple protect its users from potential harm caused by security breaches.

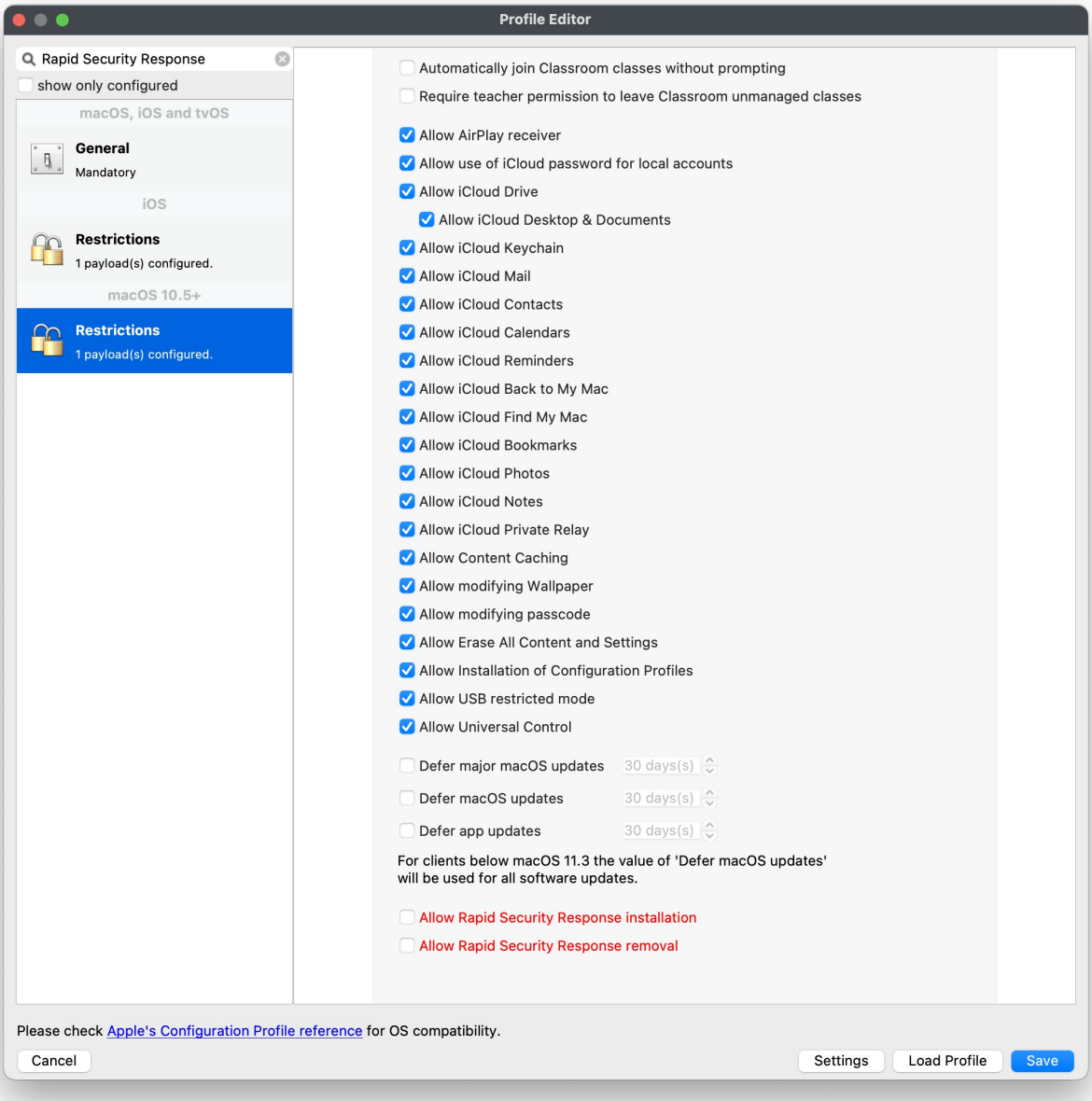
Rapid Security Responses don’t adhere to the managed software update delay; however, because they apply only to the latest minor operating system version, if that minor operating system update is delayed, the response is also effectively delayed. If necessary, the user can also remove the responses.

If a device is using the latest operating system and there is a Rapid Security Response available, AvailableOSUpdates returns the response. The MDM sends a command to install the response. Note that an MDM can only install the response on devices using the latest minor version.

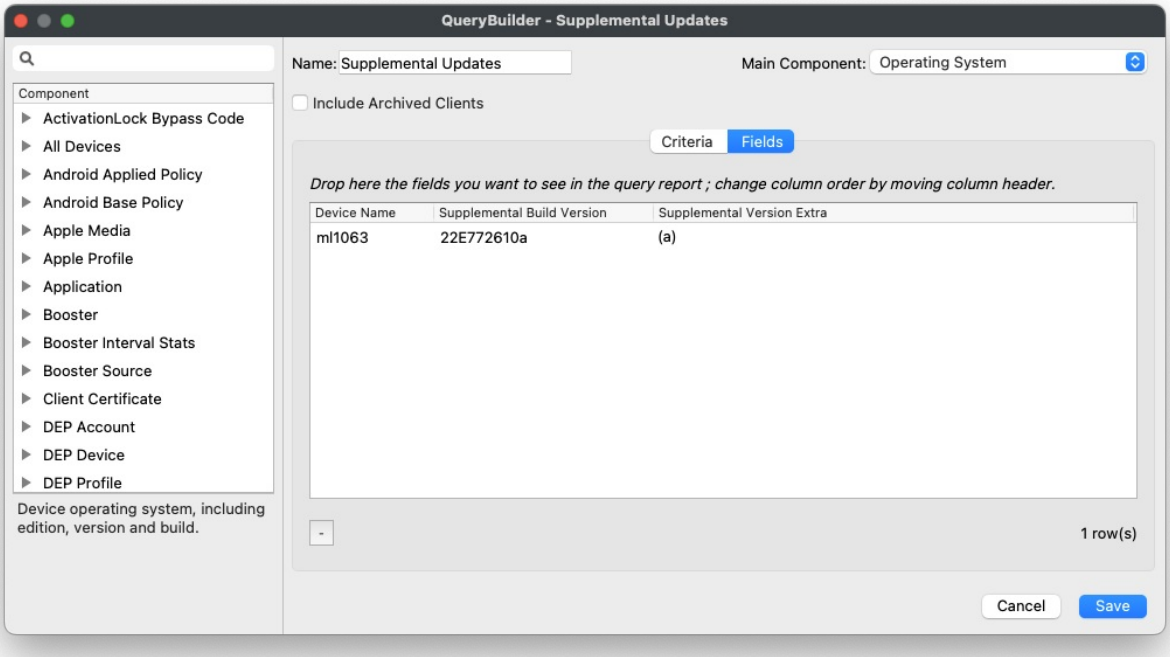
How

RSR works by leveraging Apple's resources, such as configuration profiles. There are options within macOS and iOS/iPadOS Restrictions payload to allow the installation and removal of these Rapid Security Response updates. Screen shots below for reference:


Profile Editor
Rapid Security Response
show only configured
macOS, iOS and tvOS
General
Mandatory
iOS
Restrictions
1 payload(s) configured.
macOS 10.5+
Restrictions
1 payload(s) configured.
Allow sending diagnostic and usage data to Apple
Allow modifying diagnostics settings (Supervised devices only)
Allow Touch ID / Face ID to unlock device
Allow user to unlock iOS device using an Apple Watch (Supervised devices only)
Allow password AutoFill (Supervised devices only)
Require Face ID authentication before AutoFill (Supervised devices only)
Force Apple Watch wrist detection
Allow pairing with Apple Watch (Supervised devices only)
Require passcode on first AirPlay pairing
Allow connecting to unmanaged Wi-Fi networks (Supervised devices only)
Allow setting up new nearby devices (Supervised devices only)
Allow proximity based password sharing requests (Supervised devices only)
Allow password sharing (Supervised devices only)
Allow AirPrint (Supervised devices only)
Disallow AirPrint to destinations with untrusted certificates (Supervised devices only)
Allow discovery of AirPrint printers using iBeacons (Supervised devices only)
Allow storage of AirPrint credentials in Keychain (Supervised devices only)
Allow predictive keyboard (Supervised devices only)
Allow keyboard shortcuts (Supervised devices only)
Allow keyboard auto correction (Supervised devices only)
Allow keyboard spell check (Supervised devices only)
Allow keyboard definition lookup (Supervised devices only)
Allow QuickPath typing (Supervised devices only)
Defer software updates for 30 day(s) (Supervised devices only)
Allow Apple personalized advertising
Allow NFC (Supervised devices only)
Allow Mail Privacy Protection (Supervised devices only)
Allow putting into recovery mode from an unpaired device (Supervised devices only)
Force dictation to be limited on device only (Supervised devices only)
Force translation to be limited on device only
Allow Rapid Security Response installation
Allow Rapid Security Response removal
Please check Apple's Configuration Profile reference for OS compatibility.
Cancel
Settings
Load Profile
Save



As of FileWave 15, there are two additional Inventory Items relating to Rapid Security Response; both labelled as Supplemental:



Note, Supplemental Build Version will only show a value if there is a current RSR installed on a device. Once a device updates to the next macOS version, that has no RSR installed, these two inventory items will become blank again.

Related Content

For Apple documentation regarding Rapid Security Response Updates:

- Manage Rapid Security Responses on Apple devices
- Use MDM to deploy software updates to Apple devices

FileWave and AutoPkg

Software updates that come from Apple and Microsoft are sufficient for keeping operating systems up-to-date; but there is a need to maintain currency for 3rd party applications. AutoPKG is an automation framework for macOS software packaging and distribution, oriented toward the tasks one would normally perform manually to prepare third-party software for mass deployment to managed clients. It can be thought of as an automated version of Fileset Magic for software that runs on macOS.

FileWave administrators can merge the FileWave Admin and AutoPkg environments together through a series of recipes.

Instructions for working with AutoPkg is here: https://github.com/autopkg/filewave

The primary site for AutoPkg is here: http://autopkg.github.io/autopkg with additional information here: https://github.com/autopkg/autopkg

Last, but not least, AutoPkgr is a free Mac app that makes it easy to install and configure AutoPkg. You can get that here: http://www.lindeGroup.com/autopkgr

macOS Erase and Install Fileset (Erase Optional)

Description

Upgrading or refreshing macOS from erasing requires the macOS Install App. The following provides a Fileset to handle either of these cases

> ⚠ Historically, two KBs were provided to handle each of these options. This newer Fileset allows for activation of either from one Fileset. Certain fundamental changes have been made, for example, the hidden upgrade file in /var/db is no longer required.

Ingredients

- Custom Field – "macos_instal_flag"
- Full Apple macOS Installer App
- Provided Recipe
- Provided Custom Field

Fileset:

↓ macOS


Custom Field:

↓ macOS


Full installer
Confirm the installer app is the 'Full' installer before continuing. If the app is relatively small e.g. 50MB, re-download the Install macOS Monterey.app. A full installer may be downloaded through Terminal on macOS 10.15+ devices. Example to download the 12.0 full installer to the Applications directory:

```
softwareupdate --fetch-full-installer --full-installer-version 12.0
```

Typically Apple only provides the latest point release of each Major OS version: 10.13.6, 10.14.6, etc. For a full list of available updates, use the following command:

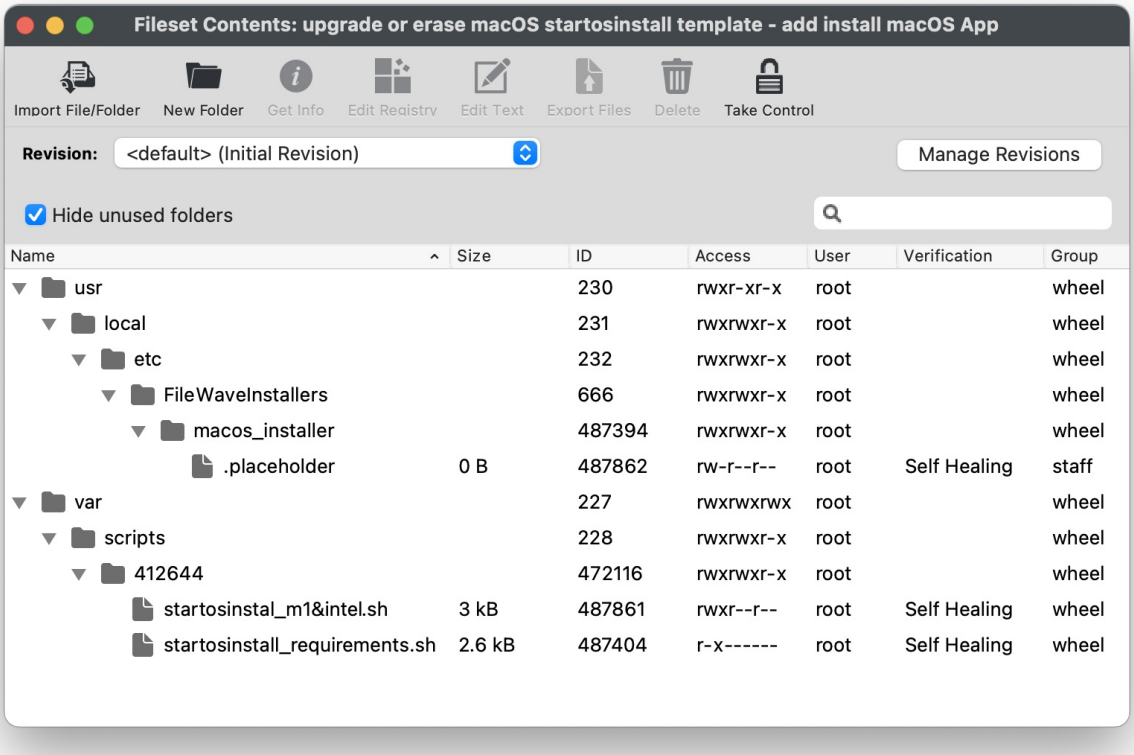
```
softwareupdate --list-full-installers
```

Directions

Fileset

1. Download and import the provided Fileset
2. Download the desired version of the macOS Installer App
3. Add the macOS Installer App into the same folder as the .placeholder fie; the .placeholder file may be removed if desired
4. Select the Instal macOS App > Get Info > Verification, select Ignore At Verify and Apply to Enclosed
5. Edit the Environment Variables for both the Activation Script and the Requirements Script

Fileset Contents

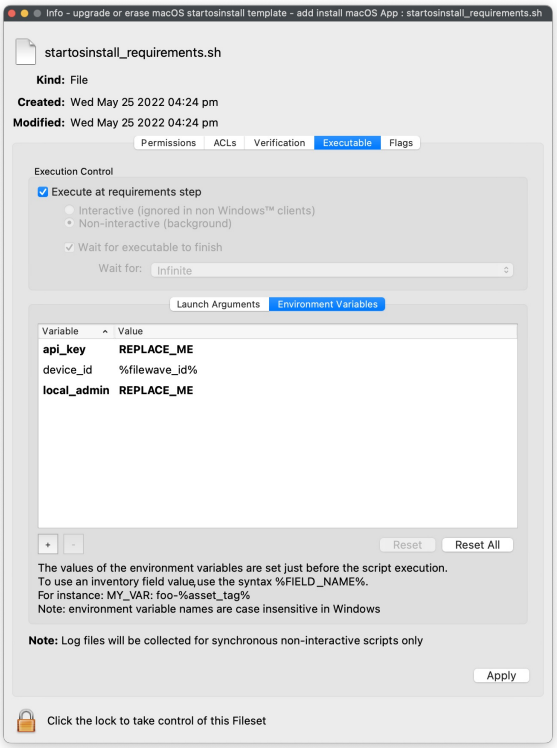


Environment Variables

The Environment Variables must be edited, providing appropriate values

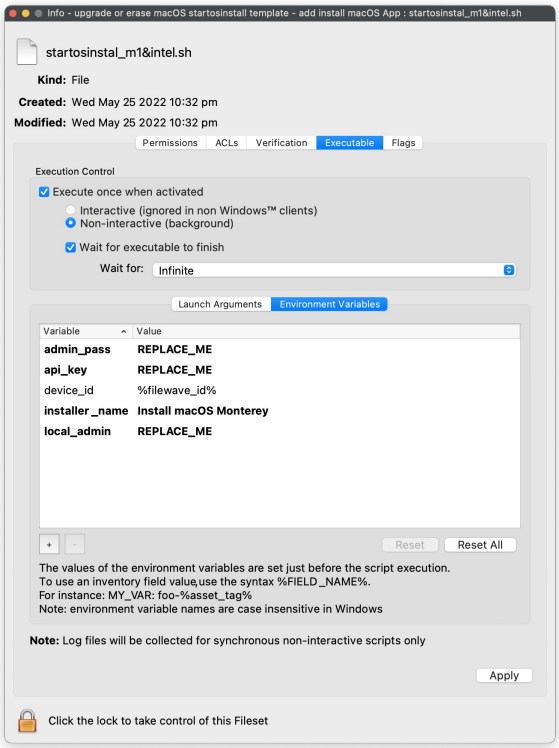
- M1 devices provide an additional complication when choosing to script the installation of macOS Instal Apps. In this instance, a local macOS Administrator Username and Password must be passed to the script. The two Variables 'local_admin' and 'admin_pass' should have their Values adapted respectively. Intel device has no such consideration and will ignore these values if set.

Requirement Script – statosinstall_requirements.sh
Key values that require editing (in bold):



- api_key – Copy a Token (base64) from a chosen FileWave Administrator's 'Application tokens'

Activation Script – startosinstal_m1&intel.sh
Key values that require editing (in bold):

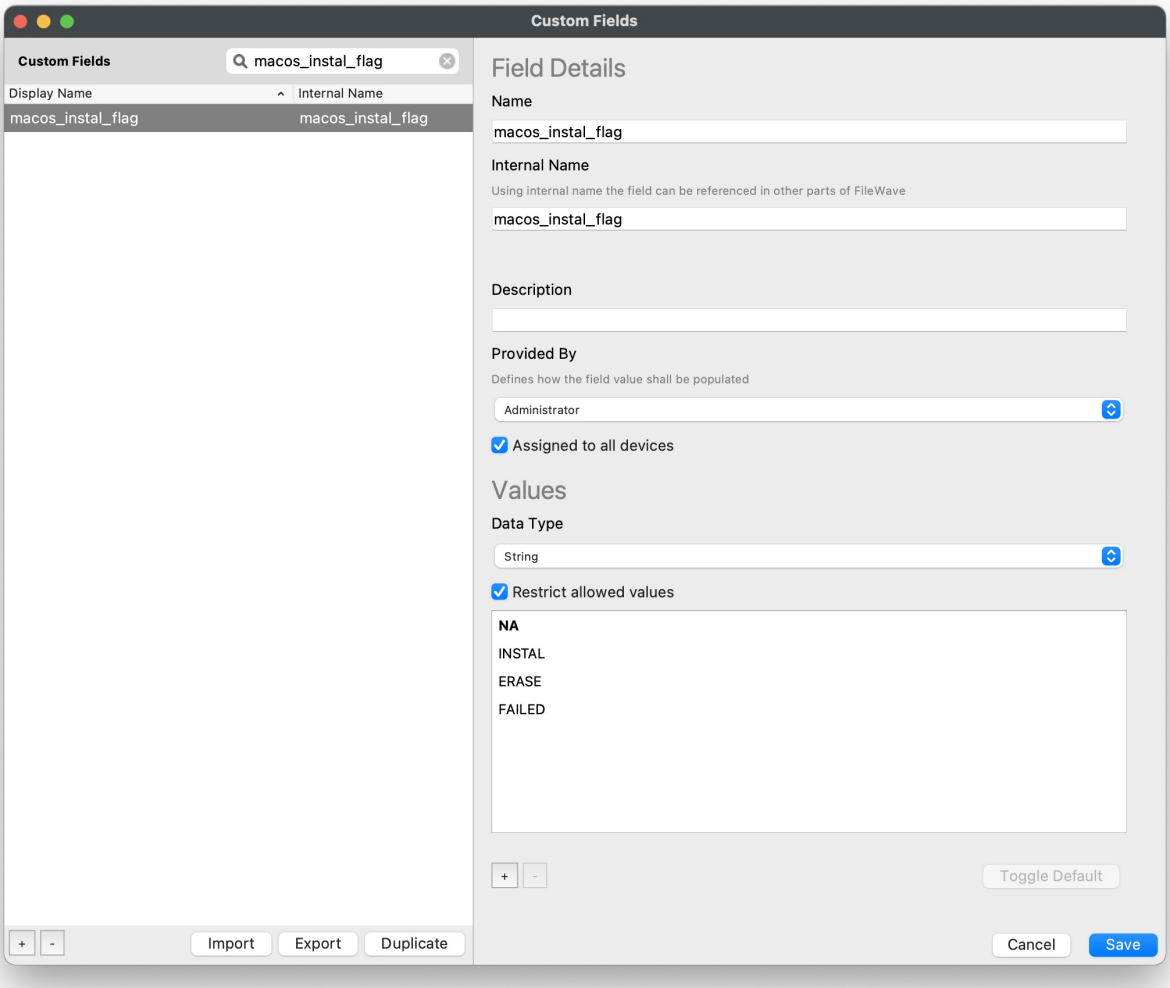


- admin_pass – password for a local admin on the target macOS device (Apple M1 only)

• local_admin – username for a local admin on the target macOS device (Apple M1 only)	• local_admin – username for the matching local admin as password on the target macOS device (Apple M1 only) • installer_name – the name of the App added, e.g. 'Install macOS Big Sur', 'Install macOS Monterey' • api_key – Copy a Token (base64) from a chosen FileWave Administrator's 'Application tokens' 'installer_name' in Fileset is also set as REPLACE_ME, but the value is left as 'Install macOS Monterey' as an example in the Fileset image.

Custom Field

Custom Field has 4 options:



- NA – default value, the Fileset will not do anything with this setting
- ERASE – the device will have the OS erased and install the provided version of macOS
- INSTAL – the device will upgrade to the provided version of macOS
- FAILED – the device will update the Custom Field flag to FAILED if something unexpected occurs. No further attempts to run the Fileset will action anything whilst this is still the case

Either 'Assign to all devices' or select appropriate devices to associate the Custom Field.

> Consider making Smart Group associations based upon the Custom Field value

Activation

Create a Smart Group based upon macos_instal_flag value being either ERASE or INSTAL

Set the associated Custom Field for a device to one of these two values depending upon the experience you would desire.

Selecting a group of devices to associate and/or alter the Custom Field value, will action that new value for all devices currently within that group. The video shows an example of setting the Custom Field value as INSTAL, with the 2 devices within the macOS 10.13 group.

Using a group in this way will have no impact on the Custom Field values for devices leaving or joining the group afterward.

Since it is likely a subsequent installation attempt will be desired upon failure (once the reason for failure has been addressed, e.g. not enough disk space), a Smart Group could be set with the Custom Field value of FAILED if desired.

To re-run a failed attempt, after addressing the reason for the failure (e.g. freeing up disk space), reset the Custom Field value to the appropriate ERASE or INSTAL value and choose to 'Reinstall Fileset'

Notes

That seemed quite complex, why?

Whilst the Fileset is associated it may trigger an action. When a device is erased, for example, any reference to the installation taking place is now lost. The consequence of this is that the device will attempt to reinstall the Fileset over and over again. Custom Fields are rescuing the process, by ensuring that once the process has been completed, there will be no re-attempt to upgrade or erase the device again since the association will no longer be in place.

By creating a FAILED Custom Field value, devices are also automatically reporting any such failure, and queries or groups may be pre-built to observe and address any such experience.

By setting the macOS Installer App as leave behind, if the installation has not been completed, but is no longer associated, this will ensure the installer remains on the device for subsequent attempts. Additionally, since Apple automatically removes the macOS Installer App after an upgrade, if the Fileset were still associated there would be no re-attempt to download the installer.

Filesets that error will automatically re-attempt installation as standard. The requirement script is designed to report an exit code of 210 to overrule this. Even if the Requirement Script fails, it will report success and will neither continue download/activation nor will it re-attempt without intervention. In this case, the client will update the Custom Field instead to report the failure. The script log should show the output of the command.

Example log message:
/usr/local/etc/macos_instal.log

|main|CUSTOM|CLIENT|startosinstall_requirements Exiting. Flag set as:

In this example, the Custom Field Flag reported has no value. This should either mean the Custom Field is not associated with the device or the script has failed to read the value. If the latter, check the Environment Variables to ensure they are correct.

Nudge for macOS Software Updates

What

Nudge is a tool designed for macOS Big Sur 11 and later. It is a multi-linguistic application offering custom user deferrals, strongly encouraging users to self update macOS.

This macOS application helps users stay up-to-date with software updates and security patches. It is a lightweight and simple background task which periodically checks for updates. Based upon a pre-defined, minimum update configuration, on detection, the user is notified, requesting they update.

Nudge can be downloaded from the following link, however this KB includes a Fileset to handle the installation of Nudge:

GitHub - macadmins/nudge: A tool for encouraging the installation of macOS security updates.

When/Why

Keeping software up-to-date is essential for any system's security and smooth functioning. However, it can be a daunting task to manually track software updates, especially when multiple software packages are installed on the device. That's where Nudge comes into the picture. It combines the automation of software update checking with manual user intervention to ensure devices remain up-to-date.

With OS versions being kept current, Nudge helps avoid security vulnerabilities and malicious, unauthorised activity all whilst improving the system's overall performance, with OS features and bug fixes. All of which should maintain a better user experience with greater productivity.

> Note, for major software updates, an Administrator password may be required. In this case, an alternate approach to upgrade devices could be considered, as highlighted in the KB: macOS Upgrades or Erasing

> Prior to macOS 11, Software Updates could be installed either through Apple's legacy Software Update catalogues or using MDM. Consider using the legacy updates for these devices; Nudge should not be required in this instance.

How

Nudge has two main components for installation;

- Installation PKG
- Configuration file

In this example, configuration will be defined using a JSON file.

The provided Fileset includes:

- Example Configuration
- Scripted method of installation
- Scripted removal when disassociated

Nudge Notifications.fileset.zip

Name	Size	Access	User	Group	Verification
▼ Library		rwxr-xr-t	root	admin	
▼ Preferences		rwxrwxr-x	root	wheel	
com.github.macadmins.Nudge.json	7.7 kB	rw-r--r--	root	wheel	Self Healing
▼ usr		rwxr-xr-x	root	wheel	
▼ local		rwxr-xr-x	root	wheel	
▼ etc		rwxr-xr-x	root	wheel	
▼ Nudge		rwxrwxr-x	root	wheel	
logo.png	61.2 kB	rw-r--r--	root	wheel	Self Healing
▼ var		rwxr-xr-x	root	wheel	
▼ scripts		rwxrwxr-x	root	wheel	
▼ 736453		rwxrwxr-x	root	wheel	
install_nudge.sh	600 B	r-x------	root	wheel	Self Healing
uninstall_nudge.sh	1.3 kB	r-x------	root	wheel	Self Healing

Scripts

This example Fileset does not contain the installer. Instead, a script pulls the latest version from GitHub and instals that version. If a new version exists, the Fileset could be re-triggered with a reinstall Fileset, causing the software to update.

The uninstaller contains the necessary lines of code to remove each item installed when disassociated

Configuration

As suggested above, in this example Fileset is a JSON for Nudge configuration. There are two lines to immediately consider:

```
            "requiredInstallationDate": "2023-12-28T00:00:00Z",
            "requiredMinimumOSVersion": "12.7.1",
```

requiredInstallationDate	When reached the conditions of user notification are altered
requiredMinimumOSversion	If OS version is below this set value, the user will begin to receive notifications

> The frequency and handling of notifications both before and after the requiredInstallationDate are handled elsewhere within the JSON file. Unless otherwise noted, all other values are set as default.

Below is the contents of the included example JSON file:

▼ JSON File metadata

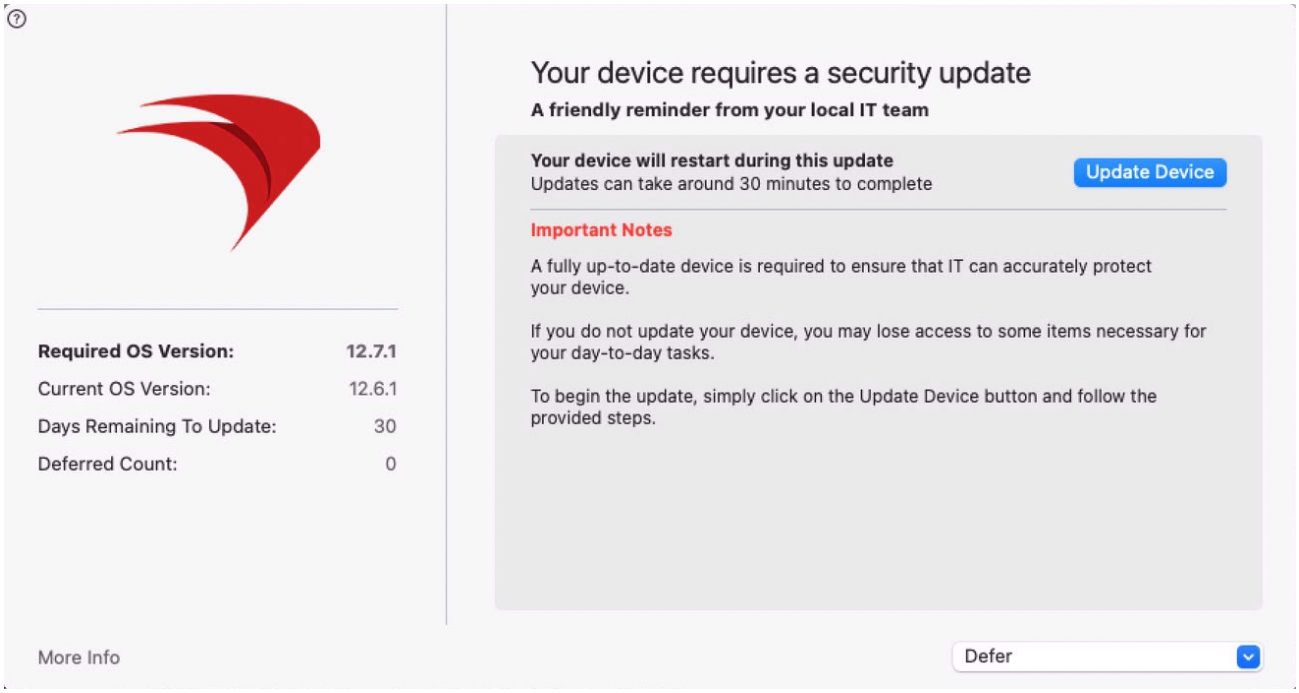

```
{
    "optionalFeatures": {
        "acceptableApplicationBundleIDs": [],
        "acceptableAssertionUsage": false,
        "acceptableCameraUsage": false,
        "acceptableScreenSharingUsage": false,
        "aggressiveUserExperience": true,
        "aggressiveUserFullScreenExperience": true,
        "asynchronousSoftwareUpdate": true,
        "attemptToBlockApplicationLaunches": false,
        "attemptToFetchMajorUpgrade": true,
        "blockedApplicationBundleIDs": [],
        "enforceMinorUpdates": true,
        "terminateApplicationsOnLaunch": false
    },
    "osVersionRequirements": [
        {
```

```
            "aboutUpdateURL_disabled": "https://support.apple.com/en-us/HT211896#macos1121",
            "aboutUpdateURLs": [
                {
                    "_language": "en",
                    "aboutUpdateURL": "https://support.apple.com/en-us/HT211896#macos1121"
                },
                {
                    "_language": "es",
                    "aboutUpdateURL": "https://support.apple.com/es-es/HT211896"
                },
                {
                    "_language": "fr",
                    "aboutUpdateURL": "https://support.apple.com/fr-fr/HT211896"
                },
                {
                    "_language": "de",
                    "aboutUpdateURL": "https://support.apple.com/de-de/HT211896"
                }
            ],
            "actionButtonPath": "/System/Library/CoreServices/Software Update.app",
            "majorUpgradeAppPath": "/Applications/Install macOS Big Sur.app",
            "requiredInstallationDate": "2023-12-28T00:00:00Z",
            "requiredMinimumOSVersion": "12.7.1",
            "targetedOSVersionsRule": "default"
        }
    ],
    "userExperience": {
        "allowGracePeriods": false,
        "allowLaterDeferralButton": true,
        "allowUserQuitDeferrals": true,
        "allowedDeferrals": 1000000,
        "allowedDeferralsUntilForcedSecondaryQuitButton": 14,
        "approachingRefreshCycle": 6000,
        "approachingWindowTime": 72,
        "calendarDeferralUnit": "imminentWindowTime",
        "elapsedRefreshCycle": 300,
        "gracePeriodInstallDelay": 23,
        "gracePeriodLaunchDelay": 1,
        "gracePeriodPath": "/private/var/db/.AppleSetupDone",
        "imminentRefreshCycle": 600,
        "imminentWindowTime": 24,
        "initialRefreshCycle": 18000,
        "launchAgentIdentifier": "com.github.macadmins.Nudge",
        "loadLaunchAgent": false,
        "maxRandomDelayInSeconds": 1200,
        "noTimers": false,
        "nudgeRefreshCycle": 60,
        "randomDelay": false
    },
    "userInterface": {
        "actionButtonPath": "/System/Library/CoreServices/Software Update.app",
        "fallbackLanguage": "en",
        "forceFallbackLanguage": false,
        "forceScreenShotIcon": false,
        "iconDarkPath": "/usr/local/etc/Nudge.logo.png",
        "iconLightPath": "/usr/local/etc/Nudge.logo.png",
        "screenShotDarkPath": "/somewhere/screenShotDark.png",
        "screenShotLightPath": "/somewhere/screenShotLight.png",
        "showDeferralCount": true,
        "simpleMode": false,
        "singleQuitButton": false,
        "updateElements": [
            {
                "_language": "en",
                "actionButtonText": "Update Device",
                "customDeferralButtonText": "Custom",
                "customDeferralDropdownText": "Defer",
                "informationButtonText": "More Info",
                "mainContentHeader": "Your device will restart during this update",
                "mainContentNote": "Important Notes",
```

```
                "mainContentSubHeader": "Updates can take around 30 minutes to complete",
                "mainContentText": "A fully up-to-date device is required to ensure that IT can accurately protect your device.\n\nIf you do not update your device, you may lose access to some items necessary for your day-to-day tasks.\n\nTo begin the update, simply click on the Update Device button and follow the provided steps.",
                "mainHeader": "Your device requires a security update",
                "oneDayDeferralButtonText": "One Day",
                "oneHourDeferralButtonText": "One Hour",
                "primaryQuitButtonText": "Later",
                "secondaryQuitButtonText": "I understand",
                "subHeader": "A friendly reminder from your local IT team"
            },
            {
                "_language": "es",
                "actionButtonText": "Actualizar dispositivo",
                "informationButtonText": "Más información",
                "mainContentHeader": "Su dispositivo se reiniciará durante esta actualización",
                "mainContentNote": "Notas importantes",
                "mainContentSubHeader": "Las actualizaciones pueden tardar unos 30 minutos en completarse",
                "mainContentText": "Se requiere un dispositivo completamente actualizado para garantizar que IT pueda proteger su dispositivo con precisión.\n\nSi no actualiza su dispositivo, es posible que pierda el acceso a algunos elementos necesarios para sus tareas diarias.\n\nPara comenzar la actualización, simplemente haga clic en el botón Actualizar dispositivo y siga los pasos proporcionados.",
                "mainHeader": "Tu dispositivo requiere una actualización de seguridad",
                "primaryQuitButtonText": "Más tarde",
                "secondaryQuitButtonText": "Entiendo",
                "subHeader": "Un recordatorio amistoso de su equipo de IT local"
            },
            {
                "_language": "fr",
                "actionButtonText": "Mettre à jour l'appareil",
                "informationButtonText": "Plus d'informations",
                "mainContentHeader": "Votre appareil redémarrera pendant cette mise à jour",
                "mainContentNote": "Notes Importantes",
                "mainContentSubHeader": "Les mises à jour peuvent prendre environ 30 minutes.",
                "mainContentText": "Un appareil entièrement à jour est nécessaire pour garantir que le service informatique puisse protéger votre appareil efficacement.\n\n Si vous ne mettez pas à jour votre appareil, vous risquez de perdre l'accès à certains outils nécessaires à vos tâches quotidiennes.\n\nPour commencer la mise à jour, cliquez simplement sur le bouton Mettre à jour le périphérique et suivez les étapes fournies.",
                "mainHeader": "Votre appareil nécessite une mise à jour de sécurité.",
                "primaryQuitButtonText": "Plus tard",
                "secondaryQuitButtonText": "Je comprends",
                "subHeader": "Un rappel amical de votre équipe informatique locale"
            },
            {
                "_language": "de",
                "actionButtonText": "Gerät aktualisieren",
                "informationButtonText": "Mehr Informationen",
                "mainContentHeader": "Ihr Gerät wird während dieses Updates neu gestartet",
                "mainContentNote": "Wichtige Hinweise",
                "mainContentSubHeader": "Aktualisierungen können ca. 30 Minuten dauern.",
                "mainContentText": "Ein vollständig aktualisiertes Gerät ist erforderlich, um sicherzustellen, dass die IT-Abteilung Ihr Gerät effektiv schützen kann.\n\nWenn Sie Ihr Gerät nicht aktualisieren, verlieren Sie möglicherweise den Zugriff auf einige Werkzeuge, die Sie für Ihre täglichen Aufgaben benötigen.\n\nUm das Update zu starten, klicken Sie auf die Schaltfläche Gerät aktualisieren und befolgen Sie die angegebenen Schritte.",
                "mainHeader": "Ihr Gerät benötigt ein Sicherheitsupdate",
                "primaryQuitButtonText": "Später",
                "secondaryQuitButtonText": "Ich verstehe",
                "subHeader": "Eine freundliche Erinnerung von Ihrem IT-Team"
            }
        ]
    }
}
```


Logo

Additionally with the Fileset is a logo file: logo.png. This file is also being referenced by the above JSON and will be seen by the user when prompted. The logo included in the example Fileset is the FileWave Logo:



The lines defining the sourced logo are:

```
        "iconDarkPath": "/usr/local/etc/Nudge/logo.png",
        "iconLightPath": "/usr/local/etc/Nudge/logo.png",
```

For a great understanding of the user experience and configuration, consider viewing the following:

MacAdmins Nudge by Neil Martin

There are more resources shown below in the Related Content and Digger Deeper section.

Deployment

After importing the above Fileset, associate with a test device. If the device is running a lower version than that defined within the JSON, a logged in user should be prompted immediately with the option to update or defer.

Once happy, consider expanding deployment until all necessary devices are included.

Reconfiguring

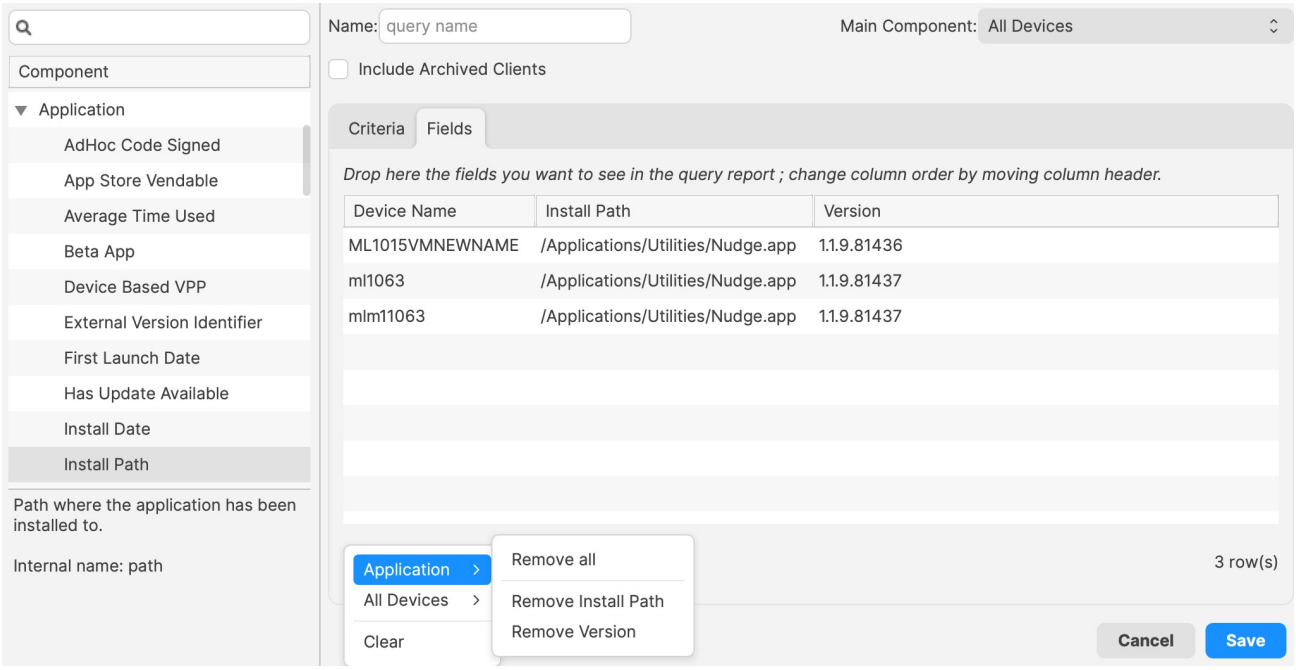
Over time it will become necessary to alter the configuration file, such that new macOS versions are set as the minimum level, with new required installation dates. Due to self-healing, this is easily handled within FileWave. Simply change those lines inside the JSON file as desired and Update Model.

Removal

If Nudge is no longer considered a requirement, disassociation of the Fileset will action the uninstaller script, which should remove all elements of Nudge from the device.

Nudge Version

As will all Applications, the version installed on devices is reported back as standard inventory. Inventory Queries could be built to observe the current version:



Related Content

- Home · macadmins/nudge Wiki · GitHub
- Getting Started · macadmins/nudge Wiki · GitHub

Digging Deeper

Customizing Nudge to meet your needs:

With Nudge, there are many more optional features and configurations that may be applied to meet your production environment. You may review these features here: optionalFeatures · macadmins/nudge Wiki · GitHub.

Apple's Rapid Security Responses:

Nudge has placed a feature-request to be added. For more information regarding, you may review the progress here: Add support for Rapid Security Response updates.

Reported Issues with macOS Software Updates

Summary

Many of our customers have reported issues related to Software Updates on macOS recently. And, we have seen issues here first-hand as well. We believe Apple are working on a number of issues related to software updates over MDM at the moment, but in the meantime, here is some information on the issues and suggested workarounds.

Software Update via MDM Overview

Software Updates for macOS since Big Sur/FileWave 14.1 have been delivered via MDM commands solely. Specifically the MDM commands AvailableOSUpdates tell us what updates are valid for a device, and once we assign an update via fileset, the download of the update is scheduled using the ScheduleOSUpdate command and DownloadOnly action. Periodically, the download progress is checked via the OSUpdateStatus command and once the device reports the update to be 100% completed, another ScheduleOSUpdate with InstallASAP action schedules the installation itself.

Experienced Issues

All in all, the above is quite simple. A standard MDM command is sent to the device to “do stuff”, and the device should do it, and we report results. But, we have seen several issues in the field such as:

- The actual macOS MDM client process stops, with an error similar to “Client crashed processing: OSUpdateStatus MDMClientError:94”
  - Behaviour: If this happens the system channel of the MDM client will no longer function, and it must be restarted
  - Workaround: See KB article here for instructions on restarting the MDM client process
- The MDM client receives a command to install a smaller update, such as XProtect or Safari, but the OSUpdateStatus command sticks at “Downloading 0%” and will not proceed
  - Behaviour: Patch does not download, nor apply
  - Workaround: Remove association for update via MDM. Use alternate method for deployment (softwareupdate -i)

Mac MDM OS Update - XProtectPayloads 89	31377	<default> (I...	31377	Downloading: 00%

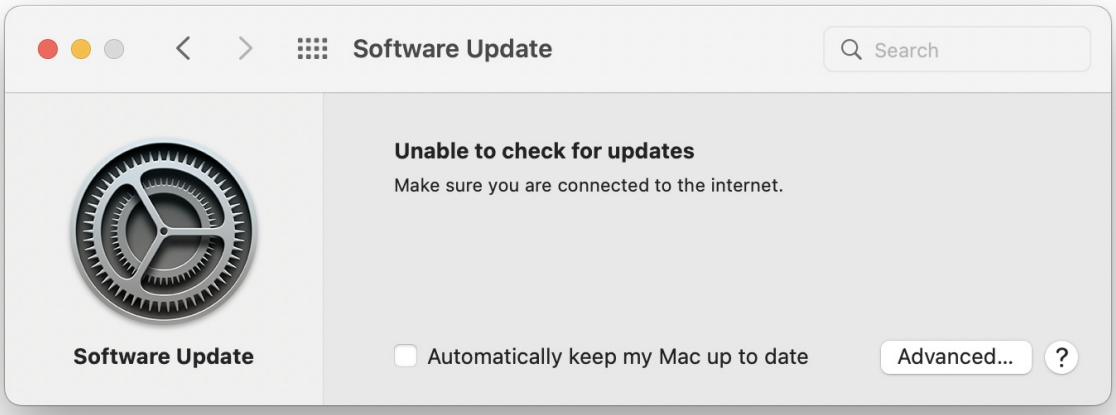
- A larger update, such as Monterey 12.6.3 is assigned, and starts to download, but fails with “Failed to personalize the software update. Please try again.”
  - Behaviour: Download begins, never progresses. Patch remains unapplied
  - Workaround: Use alternate method for deployment (softwareupdate -i)
    - Suggestion: A custom field leveraging the results of softwareupdate -l can be very useful for comparison sake
- If software update via MDM fails, and a patch is applied locally for the same, the status of the update from MDM still shows an error:
  - Behaviour: Patch is actually applied, but status is not reflected in admin UI
  - Workaround: No known workaround at moment

Name	Update ID	Size	Critical	Status
macOS Monterey 12.6.3	MSU_UPDATE_21G419_patch_12.6.3	1.62 GB	No	Error
macOS Ventura	_MACOS_13.0.1	11.3 GB	No	Unassigned
macOS Ventura	_MACOS_13.1	11.4 GB	No	Unassigned
macOS Ventura	_MACOS_13.2	11.7 GB	No	Unassigned
macOS Ventura	_MACOS_13.2.1	11.7 GB	No	Unassigned
MRTConfigData	012-04872	4.49 MB	No	Unassigned
Safari	032-38754	128 MB	No	Error
XProtectPayloads	032-38740	10.4 MB	No	Remaining
XProtectPlistConfigData	012-92414	953 kB	No	Unassigned
XProtectPlistConfigData	032-10409	954 kB	No	Unassigned

- A major upgrade such as macOS Ventura is applied to a device, but the assignment never completes:
  - Behaviour: Error message such as “Unsupported InstallAction for majorOS update” is seen in command history. This is an issue on FileWave’s part.
  - Workaround: Deploy major OS upgrade as a standard fileset using this article. We highly recommend this method regardless, as it will leverage boosters.
- You may also experience the following if checking for updates locally on the device. Despite the device being on a network and manual access to the download pages are accessible:
  - Behaviour: Unable to check for updates. Make sure you are connected to the internet.
  - Workaround: No known workaround.
    - Suggestion: The command line tool will show the following in this instance.

```
# softwareupdate -l
Software Update Tool
```

```
Finding available software
The operation couldn’t be completed. (NSURLErrorDomain error -1012.)
```



Workarounds

Please take a look at Address Stalled MDM Commands for the best workaround we've found. Please also consider opening an Apple Enterprise Support ticket as well as a FileWave Support ticket and letting us know the Apple ticket number.

You can also look at solutions like macOS Software Updates with Nudge using the JSON file method to avoid the dependency on MDM commands working.

Related Content

- Address Stalled MDM Commands
- macOS Software Updates with Nudge
- https://eclecticlight.co/2023/02/05/last-week-on-my-mac-why-are-security-updates-still-so-unreliable/
- https://eclecticlight.co/2022/08/23/has-apple-broken-content-caching-server-updates-again/
- Software Update: Where We’re Going, Where We’ve Been – Cannonball (tombridge.com)
- Addigy’s New MDM Watchdog Utility: How to Resolve MDM Issues with macOS | Addigy

S.U.P.E.R.M.A.N. for macOS Software Updates (macOS Script)

What is S.U.P.E.R.M.A.N.?

S.U.P.E.R.M.A.N. (Software Update Policy Enforcement with Recursive Messaging and Notification) can be an innovative feature within FileWave's Unified Endpoint Management (UEM) tool. Designed to optimize the macOS software updates and upgrades experience, S.U.P.E.R.M.A.N. empowers education organizations, corporations, and state and local government entities to enforce software update policies seamlessly across their diverse endpoint environments.

> Note: Sometimes this tool is referred to as super or superman.

When/Why Use S.U.P.E.R.M.A.N.?

Keeping macOS devices up-to-date with the latest software updates and upgrades is crucial to ensure optimal performance, security, and compatibility. However, managing software updates across a large number of devices can be challenging, especially in organizations with diverse endpoint environments. Here's why you should consider using S.U.P.E.R.M.A.N.:

- Streamlined Software Updates: S.U.P.E.R.M.A.N. simplifies the process of managing macOS software updates and upgrades. It ensures that all devices in the organization have the latest software versions, reducing the risk of security vulnerabilities and compatibility issues.
- Automated Policy Enforcement: With S.U.P.E.R.M.A.N., administrators can define software update policies once and enforce them across all macOS devices automatically. This automation saves time and effort while ensuring consistency in software version deployment.
- Recursive Messaging and Notification: S.U.P.E.R.M.A.N. leverages recursive messaging and notification capabilities to actively prompt users to initiate the update process. This feature encourages end-user participation in keeping their devices up-to-date, reducing potential delays in updates.
- Optimal End-User Experience: By allowing users to initiate the update process, S.U.P.E.R.M.A.N. ensures that updates don't disrupt critical tasks. End-users can conveniently schedule updates during non-productive hours, minimizing interruptions to their workflow.
- Enhanced Security and Compliance: Outdated software can expose endpoints to security risks. S.U.P.E.R.M.A.N. helps organizations maintain a secure environment by enforcing timely software updates, ensuring compliance with data protection regulations.

Mac Computers With Intel

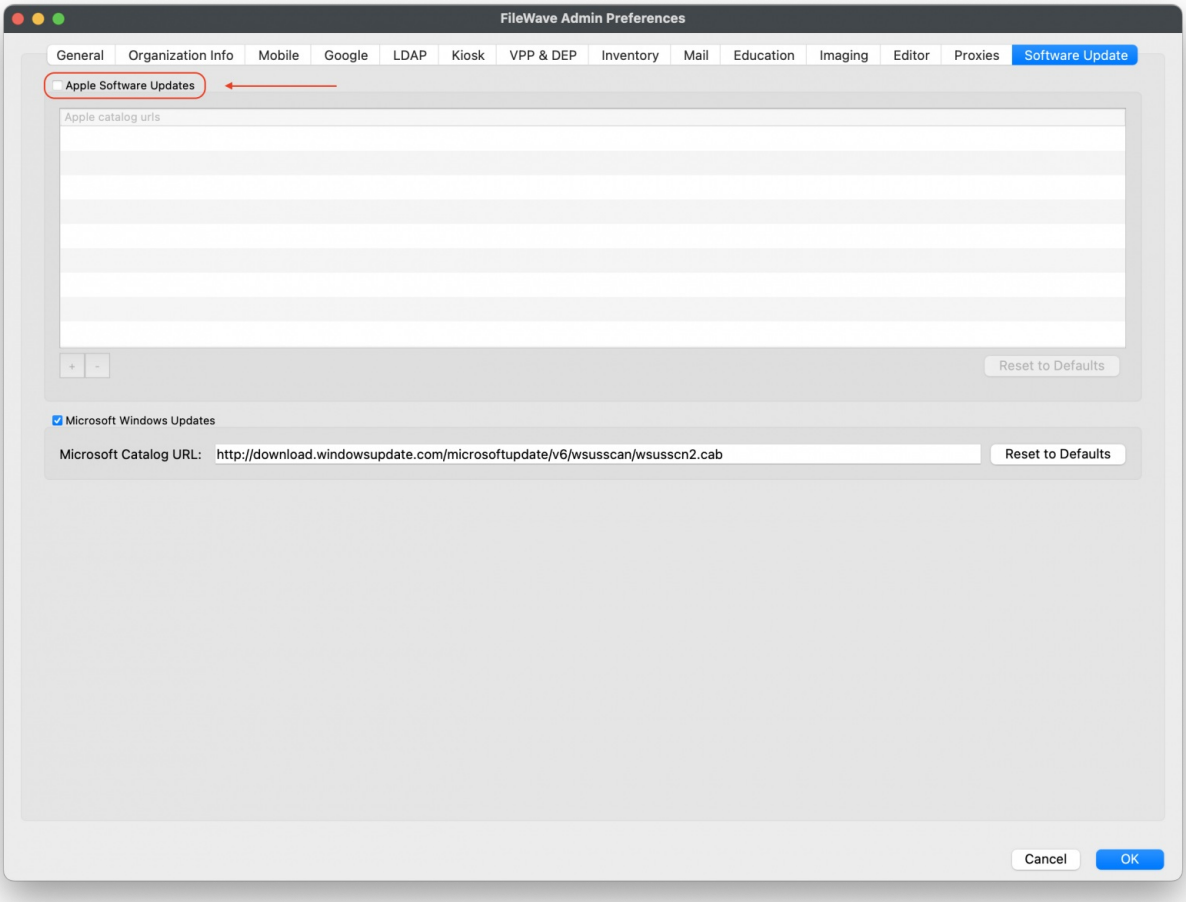
- macOS update and upgrade workflows validated on macOS 10.14 and later. Earlier versions of macOS may work, but have not been validated.
- The `super` script must run with system (root) privileges, but otherwise no additional credentials or MDM service is required.

Mac Computers With Apple Silicon

- The `super` script must run with system (root) privileges.
- To enforce automatic macOS updates or upgrades without using an MDM requires providing `super` with the local credentials of an existing account.
- If no credentials are provided to `super` then macOS updates and upgrades can not be enforced on Mac computers with Apple silicon. In this case `super` prompts the user to provide their local account password; Apple Silicon Local Credentials · Macjutsu/super Wiki · GitHub

> ⚠ Prior to macOS 11, Software Updates were pulled from Apple's catalogue (not MDM App Store) through FileWave. To achieve this, the client will overwrite the current Software Update plist whilst running and then revert the file once done. This will likely conflict with any Super process, causing Super to fail if the two occur at the same time. You can however get around this.

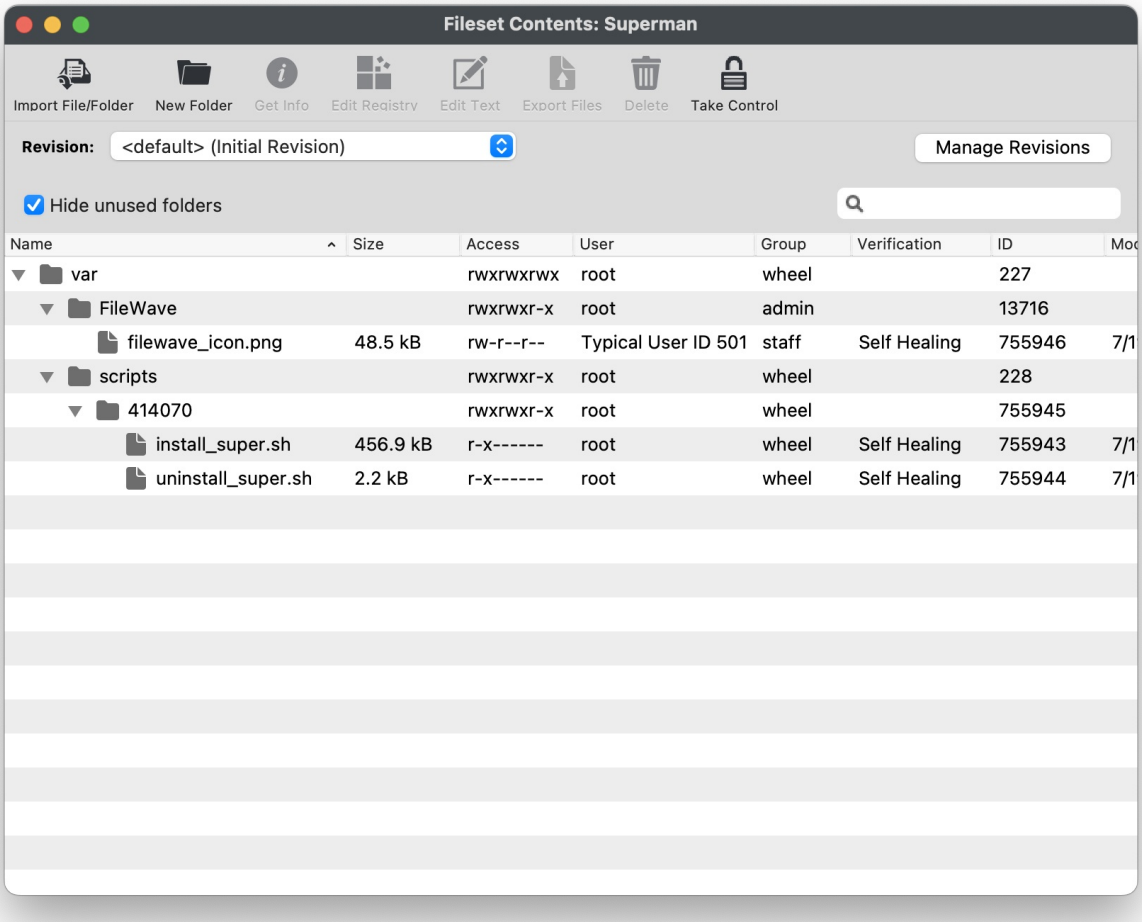
There is a client setting that could be pushed with Superpref to disable non MDM updates, but easier still, if you disable Legacy Apple Software Updates entirely from the FileWave Central > Preferences > Software Updates tab. This will allow clients not request to attempt non-MDM checks anymore.



Worth noting, that every Model Update, Inventory, Verify, the fwcld client process overwrites this file, since it needs to report which updates are appropriate, so this isn't a case of they might only conflict if you have associated updates, this happens always if not disabled.

How to Use S.U.P.E.R.M.A.N.?

The script must run with system (root) privileges, if you create the Fileset and add the `super` script, it will be run at system (root) privileges just as any other script created in FileWave Filesets.



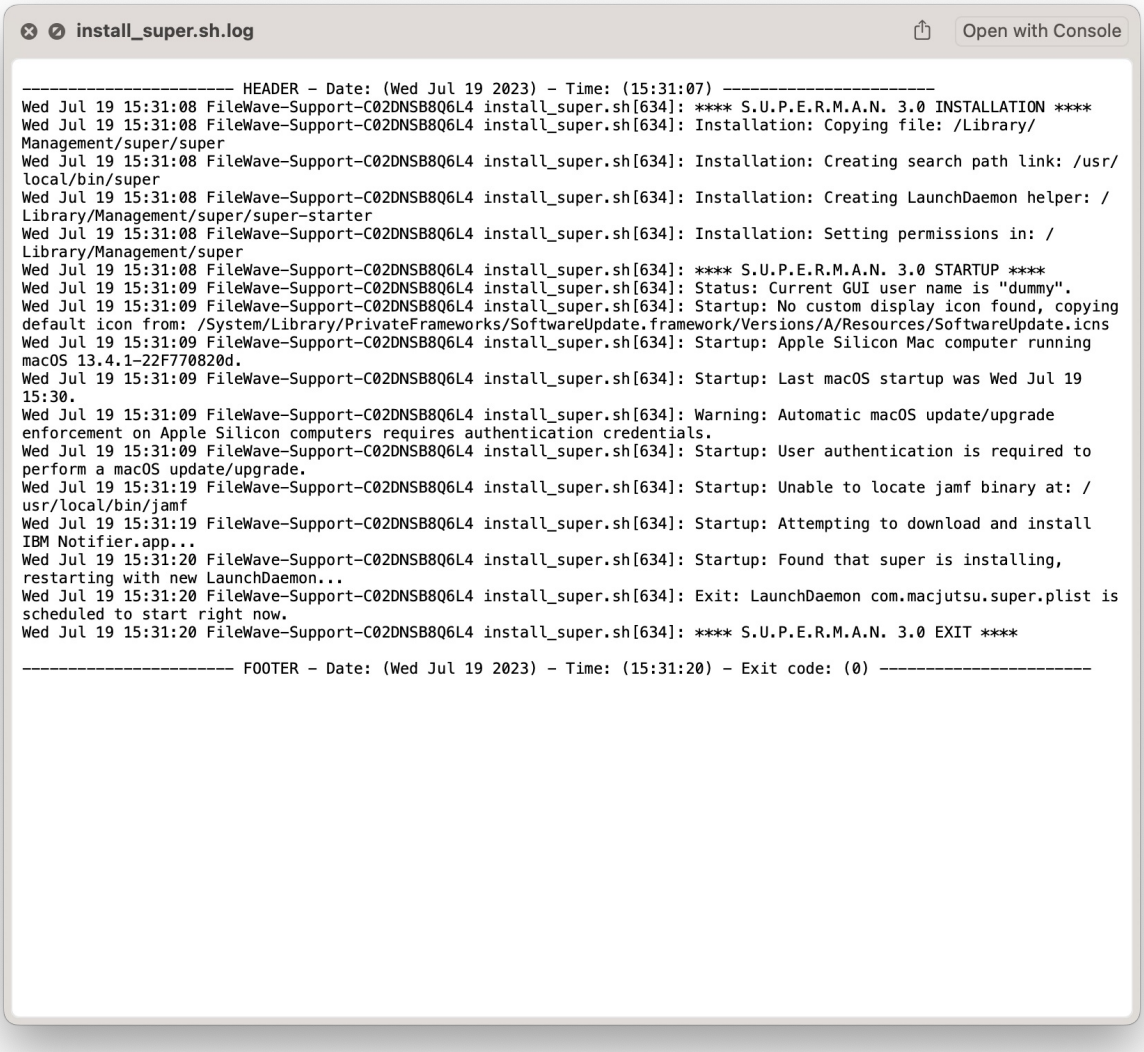
Above is the completed Fileset ready for deployment. You may also download and review the Fileset here:

Superman.fileset.zip

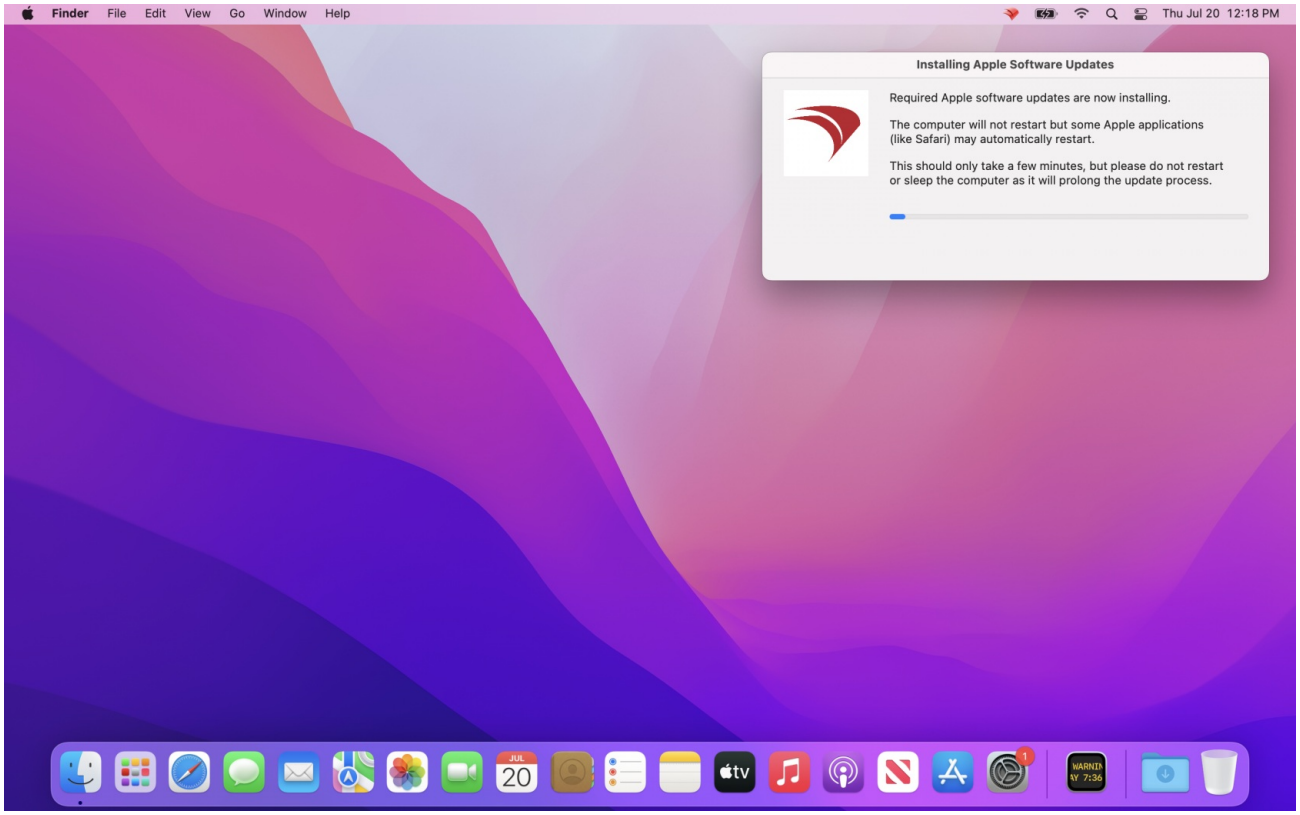
> For macOS 10.13 (Ventura) and above you should also deploy: Profile - Superman Managed Login Item.fileset.zip

The Fileset includes both an installation and uninstallation script. If you remove the Fileset from your devices, the uninstallation script will run, removing all content.

Deployment and installation will be run silently. You may review the log file to troubleshoot or confirm that the installation has been completed. The main `super` workflow log is the super.log. This log is located at /Library/Management/super/logs/super.log as set by the `$superLOG` script parameter. When run from the command line, `super` also outputs the content of the super.log in real time. Example log below:



Once installed the end user will be prompted with your configured settings. Here is an example prompt with customized icon, and enforcing the non-system updates on a macOS Monterey device:

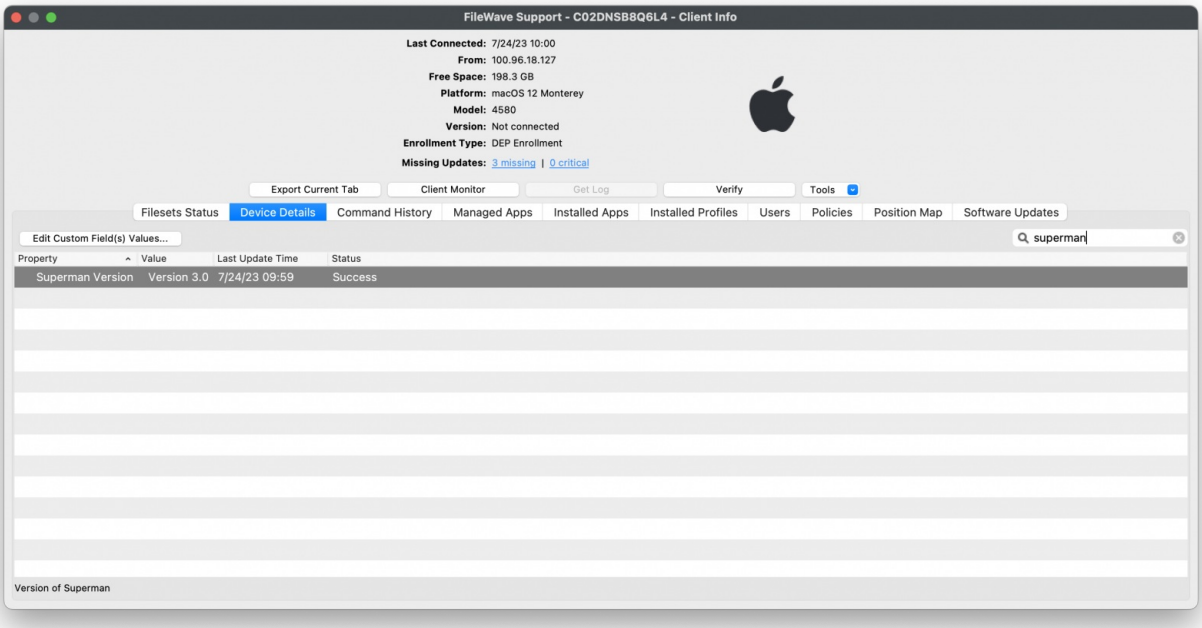


New release of S.U.P.E.R.M.A.N.

With `super` version 3.0, the default dialogs and notifications are handled by IBM Notifier.app version 2.9.1 and macOS upgrade workflows leverage `erase-install.sh` version 27.3. If either of these items are not found on the local system they are automatically installed by `super` via direct download from GitHub.

Staying up-to-date with the latest version of `super` can be managed with FileWave. Checking the version of `super`, use Custom Fields to check which version of `super` you have installed on your macOS devices. Below may you download, unzip and import the Custom Field for checking which version is installed:

Superman customfield.zip

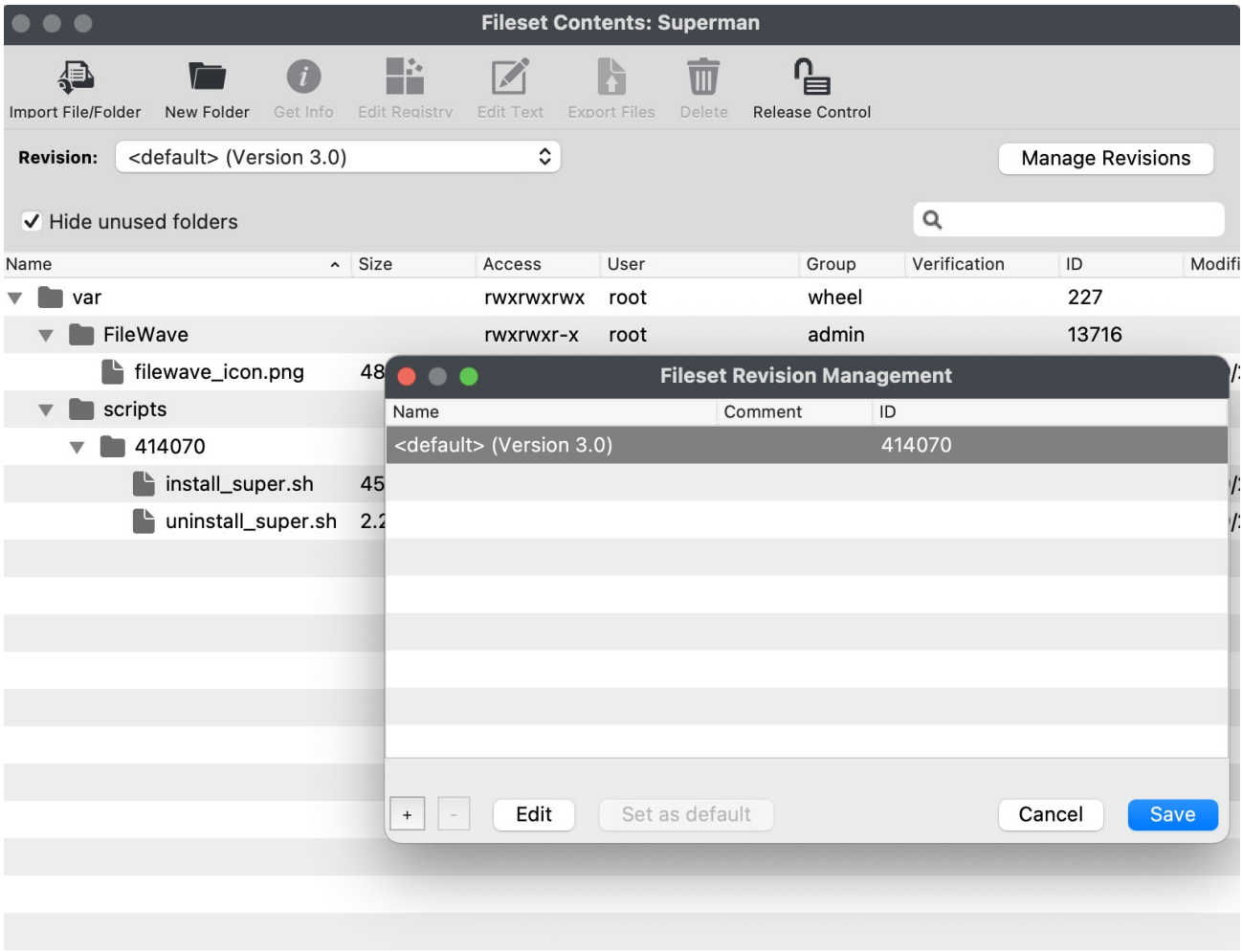


To update the Fileset and deploy the latest release of `super`, the use of Fileset Revisions works great for this Fileset.

- Highlight your `super` Fileset, then double-click to open its contents. Select Manage Revisions and create a new Fileset

Revision and choose to duplicate everything.

- Replace the install_super.sh with the newest version available from the web page: Getting Started · Macjutsu/super Wiki · GitHub. Save and test deployment to a macOS device and use the custom field to verify latest version installed.



Customizing S.U.P.E.R.M.A.N.

There are many options to customize your deployment. Displaying your customized icon for your organization, or setting up deferral dates or the amount of deferrals before software update requirement.

When customizing the icon, be sure to copy your icon into a folder within the Fileset and specific the correct location path

▼ Option Parameters

	Option parameter	Description
Allow macOS Upgrades		
	--allow-upgrade or -M	With this option enabled `super` leverages `erase-install.sh` to find compatible macOS upgrade versions. If a newer macOS upgrade is available then the `super` workflow attempts to download and install the upgrade.
Target macOS Version		
	--target-upgrade=12	With this option enabled `super` does not select any major macOS upgrades newer than the targeted version. For example, if you specified `--target-upgrade=12` then the `super` workflow never attempts to install upgrades to macOS 13 or newer.
Allow macOS RSR		
	--allow-rsr-updates or -R	If this option is enable then the `super` workflow appears

		to the user as a normal macOS update.
Enforce non-System Updates		
	--enforce-non-system-updates or -N	With this option enabled, if non-system Apple software updates are found, they are immediately downloaded and installed.
Skip All Updates		
	--skip-updates or -S	Skip checking for, downloading, or installing any Apple software updates or upgrades, even if they are available.
Display Customization		
	--display-icon=/path/icon.png	Location of the icon image, if the local path contains any special characters or spaces then you should surround the text with single `'` quotes.
Deferral Behavior		
	--focus-defer=7200	The number of seconds to defer automatically if the system is in user-enabled Focus/Do Not Disturb or when a process has requested that the display not go to sleep
	--menu-defer=300,1800,3600,7200	Display a deferral time pop-up menu in the non-deadline update restart dialog that allows the user to override the default
	--Error-defer=7200	The number of seconds to defer if `super` detects an error in the workflow
	--recheck-defer=86400	The number of seconds to defer if no software updates are available or allowed. Enabling this option results in `super` acting as a permanent agent that checks for software updates on a regular basis.
	--delete-deferrals	This option can not be set via a MDM configuration profile. However, any other deferral options that are specified via a `super` MDM configuration profile remain in effect.
Deferral Count Deadlines		
	--focus-count=5	The maximum number of automatic deferrals allowed if the system is in user-enabled Focus/Do Not Disturb or when a process has requested that the display not go to sleep
	--soft-count=5	The maximum number of user selected deferrals allowed before showing a soft deadline dialog
	--hard-count=5	The maximum number of user selected deferrals allowed before the computer automatically restarts for updates without asking the user for approval
Deferral Days Deadlines		
	--focus-days=3	The maximum number of days that automatic deferrals are allowed if the system is in user-enabled Focus/Do Not Disturb or when a process has requested that the display not go to sleep
	--soft-days=5	The maximum number of deferral days allowed before showing a soft deadline dialog.
	--hard-days=7	The maximum number of days allowed before before the computer automatically restarts for updates without asking the user for approval.
	--zero-day=2022-09-01:12:00	Instead of having the days deadline counter automatically select the day zero date, this option sets a specific date and time as day zero.

Deferral Date Deadlines		
	--focus-date=2022-09-03:12:00	The last date and time when automatic deferrals are allowed if the system is in user-enabled Focus/Do Not Disturb or when a process has requested that the display not go to sleep
	--soft-date=2022-09-05:12:00	The last date and time before showing a soft deadline dialog.
	--hard-date=2022-09-07:12:00	If this date and time have passed the computer automatically restarts for updates without asking the user for approval.
Apple Silicon Local Credentials		
	--local-account='labadmin' --local-password='ThisIs@Test'	An existing local (standard or admin) user account name and password with volume ownership privileges that can be used to authenticate the local `softwareupdate` command.

MDM Configuration Profile

If there are specific `super` options you plan to set "permanently" then you should consider deploying these settings via a MDM configuration profile. In addition to over-the-air deployment, using a MDM configuration profile also allows you to enforce your options. In other words, if a specific `super` option is deployed via a MDM configuration profile then it cannot be ignored or changed via local command options.

The MDM configuration profile specification allows for custom settings deployed via application specific preference domains. In the case of `super`, the preference domain is `com.macjutsu.super`.

> ⚠ Please note: Do not deploy the example .mobileconfig file, as this is an example for all options and contains conflicts with other settings that can cause errors if deployed as is.

Example options for the .mobileconfig listed below.

▼ Example .mobileconfig

```
<?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">
  <dict>
    <key>PayloadUUID</key>
    <string>D819E6B3-72BA-4202-874D-FE7C6783C984</string>
    <key>PayloadType</key>
    <string>Configuration</string>
    <key>PayloadOrganization</key>
    <string>Macjutsu</string>
    <key>PayloadIdentifier</key>
    <string>D819E6B3-72BA-4202-874D-FE7C6783C984</string>
    <key>PayloadDisplayName</key>
    <string>S.U.P.E.R.M.A.N. All Settings Example</string>
    <key>PayloadDescription</key>
    <string>This is an example of all possible super managed preferences. DO NOT DEPLOY. THIS EXAMPLE CONTAINS
CONFLICTING SETTINGS THAT WILL CAUSE ERRORS IF DEPLOYED.</string>
    <key>PayloadVersion</key>
    <integer>1</integer>
    <key>PayloadEnabled</key>
    <true/>
    <key>PayloadRemovalDisallowed</key>
    <true/>
    <key>PayloadScope</key>
    <string>System</string>
    <key>PayloadContent</key>
    <array>
      <dict>
        <key>PayloadDisplayName</key>
        <string>Custom Settings</string>
```

```
        <key>PayloadIdentifier</key>
        <string>22539B64-64B4-4973-AA52-B6B105B4C014</string>
        <key>PayloadOrganization</key>
        <string>Software</string>
        <key>PayloadType</key>
        <string>com.apple.ManagedClient.preferences</string>
        <key>PayloadUUID</key>
        <string>22539B64-64B4-4973-AA52-B6B105B4C014</string>
        <key>PayloadVersion</key>
        <integer>1</integer>
        <key>PayloadContent</key>
        <dict>
          <key>com.macjutsu.super</key>
          <dict>
            <key>Forced</key>
            <array>
              <dict>
                <key>mcx_preference_settings</key>
                <dict>
                  <key>AllowRSRUpdates</key>
                  <true/>
                  <key>AllowUpgrade</key>
                  <true/>
                  <key>BatteryLevel</key>
                  <string>50</string>
                  <key>BatteryTimeout</key>
                  <string>3600</string>
                  <key>DefaultDefer</key>
                  <string>120</string>
                  <key>DeferDialogTimeout</key>
                  <string>60</string>
                  <key>DisplayAccessoryDefault</key>
                  <string>https://raw.githubusercontent.com/Macjutsu/super/main/Super-Friends/Display-
Accessory-Example.html</string>
                  <key>DisplayAccessoryType</key>
                  <string>HTML</string>
                  <key>DisplayAccessoryUpdate</key>
                  <string>https://raw.githubusercontent.com/Macjutsu/super/main/Super-Friends/Display-
Accessory-Example.html</string>
                  <key>DisplayAccessoryUpgrade</key>
                  <string>https://raw.githubusercontent.com/Macjutsu/super/main/Super-Friends/Display-
Accessory-Example.html</string>
                  <key>DisplayAccessoryUserAuth</key>
                  <string>https://raw.githubusercontent.com/Macjutsu/super/main/Super-Friends/Display-
Accessory-Example.html</string>
                  <key>DisplayIcon</key>

<string>/System/Library/CoreServices/CoreTypes.bundle/Contents/Resources/BurningIcon.icns</string>
                  <key>DisplayRedraw</key>
                  <string>20</string>
                  <key>DisplaySilently</key>
                  <false/>
                  <key>EnforceNonSystemUpdates</key>
                  <true/>
                  <key>ErrorDefer</key>
                  <string>120</string>
                  <key>FocusCount</key>
                  <string>5</string>
                  <key>FocusDate</key>
                  <string>2022-07-01</string>
                  <key>FocusDays</key>
                  <string>2</string>
                  <key>FocusDefer</key>
                  <string>120</string>
                  <key>FreeSpaceTimeout</key>
                  <string>3600</string>
                  <key>FreeSpaceUpdate</key>
                  <string>15</string>
                  <key>FreeSpaceUpgrade</key>
                  <string>35</string>
```

```
                <key>HardCount</key>
                <string>5</string>
                <key>HardDate</key>
                <string>2022-07-03:03:03</string>
                <key>HardDays</key>
                <string>6</string>
                <key>HelpButton</key>
                <string>https://support.apple.com/en-us/HT201541</string>
                <key>IconSizeIbm</key>
                <string>128</string>
                <key>IconSizeJamf</key>
                <string>128</string>
                <key>InstallNow</key>
                <true/>
                <key>JamfProID</key>
                <string>$JSSID</string>
                <key>MenuDefer</key>
                <string>120,200,300</string>
                <key>OnlyDownload</key>
                <true/>
                <key>PolicyTriggers</key>
                <string>trigger1,trigger2</string>
                <key>PreferJamfHelper</key>
                <false/>
                <key>RecheckDefer</key>
                <string>120</string>
                <key>RestartWithoutUpdates</key>
                <true/>
                <key>SkipUpdates</key>
                <true/>
                <key>SoftCount</key>
                <string>5</string>
                <key>SoftDate</key>
                <string>2022-07-02:02</string>
                <key>SoftDays</key>
                <string>4</string>
                <key>SoftDialogTimeout</key>
                <string>60</string>
                <key>TargetUpgrade</key>
                <string>12</string>
                <key>TestMode</key>
                <true/>
                <key>TestModeTimeout</key>
                <string>60</string>
                <key>UserAuthMDMFailover</key>
                <string>HARD,INSTALLNOW,BOOTSTRAP</string>
                <key>UserAuthTimeout</key>
                <string>600</string>
                <key>VerboseMode</key>
                <true/>
                <key>WarningButton</key>
                <string>https://support.apple.com/en-us/HT201222</string>
                <key>ZeroDay</key>
                <string>2022-07-01</string>
              </dict>
            </dict>
          </array>
        </dict>
      </dict>
    </dict>
  </array>
 </dict>
</plist>
```

Uninstalling S.U.P.E.R.M.A.N.

The Super-Friends folder contains a `Remove-Super.sh` script that deletes all `super` related items except for the `erase-`

`install.sh` items used to facilitate macOS upgrade workflows.

However, if `erase-isntall.sh` was used as part of a macOS upgrade workflow it too can be easily removed by deleting the contents of the /Library/Management/erase-install/ folder.

The script has been included in the Fileset on this KB article as well so you can have it executed simply by removing the Association of the Fileset to your devices. If you don't want that behavior then simply remove the script from the Fileset.

Related Content

- Home · Macjutsu/super Wiki (github.com)
- Getting Started · Macjutsu/super Wiki · GitHub
- MDM profile example - Macjutsu/super Wiki

Software Updates in the age of macOS MDM (Big Sur v11.0+ / iOS 15+)

What

For many years now, FileWave has leveraged the Software Update tool on macOS devices to evaluate and to deploy software updates. With the release of macOS 11 (Big Sur) this behavior has changed somewhat.

When/Why

We'll still want to deploy software updates, and from a FileWave admin perspective, the process for assigning the updates has not changed, but the mechanism that delivers those updates behind the scenes has changed. From Big Sur onwards, all Software Updates will be delivered through MDM commands only. So, the "How" of assigning updates has not changed, but the method of deployment has.

Here are some important items to note regarding this change:

- This requires all Big Sur+ devices to be MDM enrolled to enable delivery of Software Updates
- The updates are handled more like iOS updates were already handled...i.e. the device is notified to update, and the device itself gathers the update from Apple
  - The result of the above is that Apple caching servers become even more important to have in your environment
  - The delivery method being changed to MDM eliminates FileWave boosters from caching these updates
- This change also means that restriction profiles that defer Software Updates for up to 90 days are also pertinent now for macOS

How

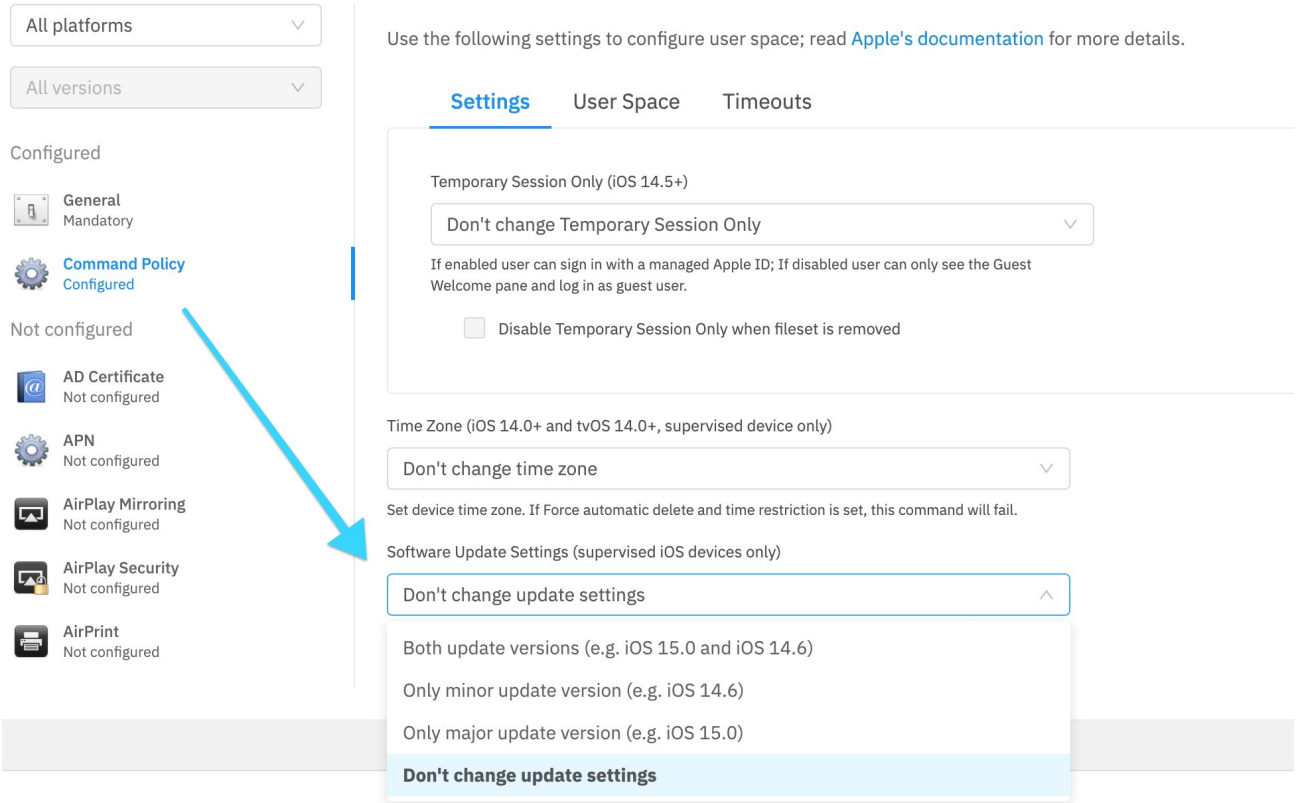
We'll still want to deploy software updates, and from a FileWave admin perspective, the process for assigning the updates has not changed, but the mechanism that delivers those updates behind the scenes has changed. From Big Sur onwards, all Software Updates will be delivered through MDM commands only. So, the "How" of assigning updates has not changed, but the method of deployment has.

Here are some important items to note regarding this change:

- This requires all Big Sur+ devices to be MDM enrolled to enable delivery of Software Updates
- The updates are handled more like iOS updates were already handled...i.e. the device is notified to update, and the device itself gathers the update from Apple
  - The result of the above is that Apple caching servers become even more important to have in your environment
  - The delivery method being changed to MDM eliminates FileWave boosters from caching these updates
- This change also means that restriction profiles that defer Software Updates for up to 90 days are also pertinent now for macOS

macOS 12 devices now report the right information to properly use the Apple Software Update Lookup Service. Specifically, the ProductVersion key is now supported for macOS (allows definition of update version to install) and FileWave will be sending 2 InstallSoftware commands as per Apple's direction in order to make software updates happen in a more timely fashion.

Starting with iOS 15, it will be possible to stay on the previous version (today, iOS 14) and still get minor updates. A new Settings command allows to define if the end-user will be presented the most recent update (iOS 15), the oldest (iOS 14.x), or both options as seen below in the updated Command Policy profile.



Related Content

- Apple Software Updates - macOS
- Software Updates: Deploying to Groups

Troubleshooting dead MDM Client on macOS using launchctl kickstart Command

As a warning the workaround listed in this article has been identified as not being recommended by Apple. There are multiple Internet discussions on various MDM forums saying this worked for them so we wanted to publish this to offer an option, but please do contact Apple to become aware of an official solution.

What

MDM (Mobile Device Management) is an essential tool for managing devices in an organization, and sometimes MDM client on macOS systems can stop processing commands. If you're experiencing this issue, let us know through support, but as a workaround, you can use the following to get things flowing again. Because FileWave uses a hybrid approach, we can do this workaround even when the OS built-in MDM client has become stuck.

```
sudo /bin/launchctl kickstart -k system/com.apple.softwareupdated
```

In this article, we'll walk you through what this command does, when and why you should use it, how to use it, and provide related links. We've also included a Fileset that uses this command in a Verification Script so that it will run once per day. You should only use this workaround until a future OS update addresses this issue. At that time, you can remove the Association to the Fileset since the workaround won't be needed anymore.

The launchctl kickstart command is used to start system services on macOS. The kickstart subcommand is used to start a system service, and the -k option tells kickstart to send a stop signal to the specified service before starting it.

When/Why

If the MDM client on your macOS system is not processing commands, you can use the described command to restart the softwareupdated system service. This service is responsible for checking for and downloading updates for macOS and other Apple software. This command can be used as a workaround until the issue is addressed in a future OS update. In looking at this issue it is possible that an OS update is contributing to the issue so you may want to clear out OS updates from FileWave and have the most recent ones enabled. Smaller updates like Safari may be more problematic. We're working on what we can do to make this not an issue.

How

To use the `launchctl kickstart` command, open the Terminal application and type the following command:

```
sudo /bin/launchctl kickstart -k system/com.apple.softwareupdated
```

This will restart the softwareupdated system service, which is responsible for checking for and downloading updates for macOS and other Apple software. You will need to enter your admin password to run this command as sudo.

Using Fileset for launchctl kickstart command:

We have created a Fileset that includes a Verification Script which uses the launchctl kickstart command to restart the softwareupdated system service once per day. This Fileset can be used as a workaround until Apple addresses the MDM client issue. You can download the Fileset from our website and then associate it with the macOS devices that are experiencing the MDM client issue.

Digging Deeper

If you want to learn more about how the `launchctl kickstart` command works, you can refer to the `launchctl` manual page by typing the following command in Terminal:

```
man launchctl
```

This will provide you with detailed technical information on `launchctl` and its subcommands, including `kickstart`. You can also refer to the Apple Developer website for more information on the macOS system services and how they work.