Uninstalling Filesets Safely

Plan removal before deployment

A reliable Fileset deployment includes a removal plan. Uninstall behavior depends on the platform, package type, vendor uninstaller, self-healing settings, and whether the Fileset is being removed permanently or replaced by a newer revision.

Use this guide to decide what should happen when a Deployment is removed. Older FileWave interfaces may describe the same transition as disassociation.

Mobile Devices

On iOS and Android, removing an app or configuration Deployment normally removes that managed item from the device. The surrounding data and connectivity impact still needs review.

App data may be stored inside the app. Unless the app uses cloud storage or another documented persistence method, removing it can also delete the user's local app data.

Before removing a configuration, check whether it provides network access, certificates, VPN settings, or another dependency. Removing that configuration can also remove the device's path back to FileWave.

Computer Devices

Computer installers can be supplied in multiple methods, including PKG, MSI, EXE, standard file-level Filesets, scripts and registry entries. As such, the topic of uninstalling is somewhat larger.

Ideally, developers will supply some kind of uninstaller, be that another PKG, MSI or EXE or a supplied script. However, this is often not the case.

Generic Filesets

Where Filesets are set to deliver files with installers, self-healing can therefore supply the necessary requirements to ensure files are removed on disassociation. Note though, that during use, applications may create additional files throughout the file system. For a complete removal, if the developer of the App does not supply a pre-configured uninstaller, then these files would need to be identified and also removed.

> ⚠ Note, just because a vendor has provided an uninstaller, does not necessarily mean it is a full uninstaller and some files may still be left behind.

FileWave has a couple of ways that these could be dealt with, once identified.

Script

A script could be created to run removal commands against each of these files or folders

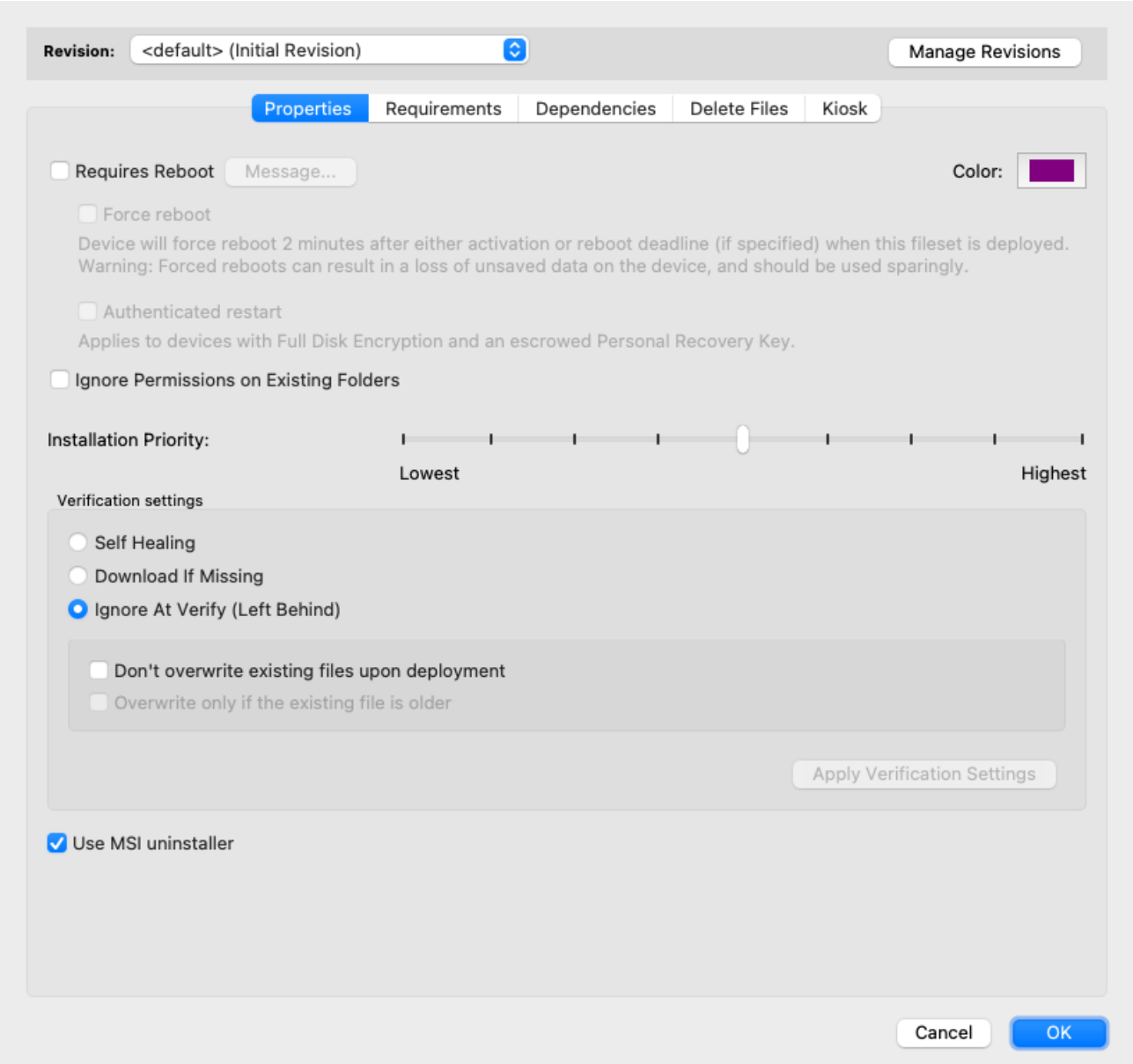
Fileset Files

An additional Fileset could be built to include these extra files and would be set as download if missing. Download if missing does not overwrite files if they exist (even if they differ), but will remove files on dissociation.

Windows

MSI, EXE and registry.

MSI packages can remove installed files by product identifier. For an MSI-specific Fileset, enable Use MSI uninstaller in Fileset properties when removal of the Deployment should run the MSI uninstall action.



> ⚠ Packaging matters. If you place an MSI inside a standard Fileset and install it from a script, FileWave does not treat that Fileset as an MSI package. The Use MSI uninstaller option therefore cannot perform the removal; provide an explicit uninstall script instead.

It is possible to script the removal instead, by way of the uninstall string. The uninstall strings are stored in the register and can be found using the following PowerShell script. The example is set to return installers with FileWave in their name:

```
# Edit the name of the string to search
$Display_Name_Search = "FileWave"

Get-ChildItem -Path HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall, HKLM:
\SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Uninstall | Get-ItemProperty |
Where-Object {$_.DisplayName -match $Display_Name_Search } | Select-Object -Property
DisplayName, UninstallString
```

The output might look something like:

```
DisplayName     UninstallString
-----------     ---------------
FileWave Admin  MsiExec.exe /I{A7182B5E-C5DA-449B-9060-490C5DBC878F}
FileWave Client MsiExec.exe /I{D9344A9A-7212-40DB-AC5B-131308325104}
```

One uninstall string for FileWave Admin and another for FileWave Client. The uninstall string may then be used in a Fileset PowerShell Script.

EXE on the other hand, do not have such an option. Again, if not developer supplied, one of the above generic methods would need to be considered.

Likewise, registry entries would require a method for removal if required, during a disassociation of the Fileset.

macOS

PKG installers have no built-in uninstaller. In some instances, vendors will supply an uninstaller, but all of the considerations mentioned in the generic section would need to be considered.

How to Uninstall

Not so much how, but when.

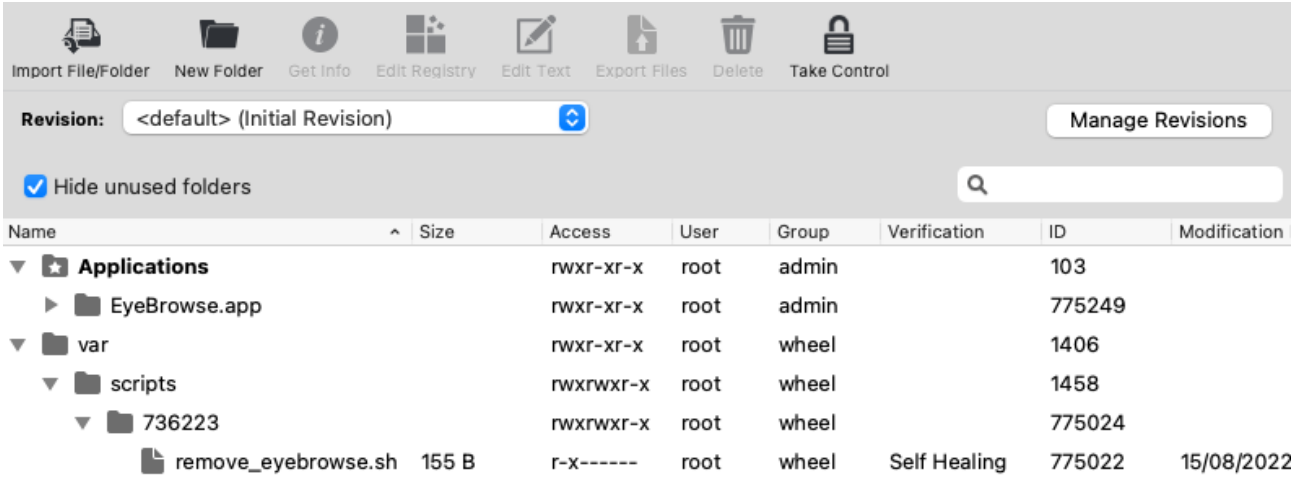
Many example recipes provide an uninstaller within the Fileset. As such, if the Fileset is disassociated, the uninstaller will trigger. However, this may not be ideal.

For example.

Problem

Imagine you have a self-healing Fileset for the new imaginary web app, EyeBrowse version 1.0. Let's assume, that this adds additional files in the user environment when ran. The following might be actioned:

- Identify additional files
- Script the removal
- Add an uninstaller script within the same Fileset



Now consider the release of version 1.2.

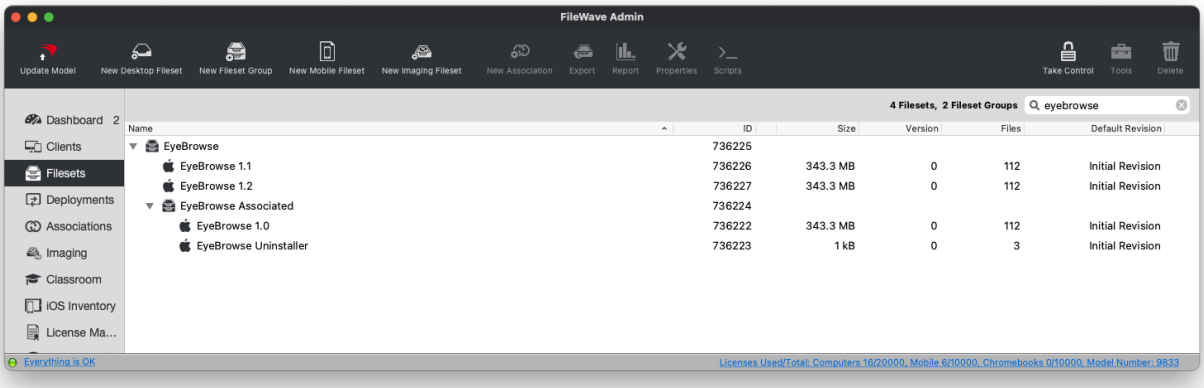
- Create a new Fileset, revision or duplicate the original
- Edit this Fileset for the version 1.2

When transitioning between these two versions, the original Fileset should become disassociated. However, when this occurs, the uninstaller script will run, removing the identified, user specific files. This is likely to be an unwanted outcome, since the user is still using the software, the intention is for them to have a newer version.

Resolution

What would be better is to create a Fileset Group. The group would require:

- The EyeBrowse App Fileset (holding just the installer and any configuration)
- The additional user data removal uninstaller Fileset
- The Fileset Group is associated to devices ('EyeBrowse Associated')



With the above configuration, when changing between version 1.0 to 1.2 of the App, the EyeBrowse App Fileset would be swapped out of the Fileset group or the revision altered. If the App and all associated files needed to be completely removed, at this point the Fileset Group would be disassociated, which in turn would also cause the uninstaller script to run, from the additional, contained Fileset.

When downloading recipe Filesets with uninstaller included, consider if it is desirable to separate the installer from this Fileset. If so, consider duplicating the supplied Fileset, placing both into one Fileset Group, and removing the installer from one and the uninstaller from the other.

In some instances, it may be desirable to run the uninstaller on each new update, providing a clean instance. Each Fileset should be considered in its own right.

Updating Uninstallers

Installers are forever changing, with point and major releases being offered frequently. This may of course mean Uninstallers also require updating. For vendor supplied ones, check for newer supplied versions, but for self built methods, these should be checked and updated if need be.

Some installers can be downloaded and actioned all in one process. Brew has an example of this:

https://github.com/homebrew/install#uninstall-homebrew

For example:

```
NONINTERACTIVE=1 /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/
Homebrew/install/HEAD/uninstall.sh)"
```

As such, an uninstaller Fileset script could be created, which, in theory, could just include this line.

This is both advantageous and also troublesome.

Pros

The uninstaller actioned will aways be the latest version

Cons

If the version currently installed is not the latest version, but one or move versions older, the script may no longer be appropriate

Consideration

If the uninstaller is downloaded at the same time as the installer, then the two should align. A Fileset uninstaller script could be built to contain the contents of this script and as such, should be the correct uninstaller for the active Fileset. With each update, the uninstaller could also be compared.

Note, there could still be extra checks. Since FileWave runs as 'System' on Windows and 'root' on macOS, any vendor supplied script should be checked for user specific paths and how these paths are addressed, for example.

Revision #3
★Created 2023-07-24 10:20:33 UTC by Sean Holden
Updated 2026-08-07 12:55:52 UTC by Josh Levitsky