VPP App Updates for macOS / iOS / tvOS devices

Description

As standard, VPP Apps on devices should update automatically, regardless of how they were installed, e.g. Kiosk or Standard Deployment. They may also be occasion to block VPP Updates, without locking the entire device.

Information

By default, with no defined customisation, FileWave will trigger automatic updates of VPP Apps installed on devices.

At certain times, FileWave Server requests the list of installed applications. MDM commands to devices may be observed in the 'Command History' tab of a device's information:

ManagedApplicationList	acknowledged	2024-09-23T11:52:27	2024-09-23T11:52:28
InstalledApplicationList	acknowledged	2024-09-23T11:52:18	2024-09-23T11:52:27

At minimum, this will occur every Automatic Verify (usually 24hrs), assuming devices are online, but other actions should also trigger these events. For example:

- Manual Verify
- Model Update
- Smart Group changes
- Opening Client Info for a device

If an installed App has an update, that App will be flagged.

Where an App has an update Flag, a new command to instal the application is queued with the device. This should be true for all Apps that have an update, one entry per App. On receipt of the request, the device communicates with the App Store and acknowledges the request to update back to the FileWave Server.

InstallApplication	acknowledged	2024-09-18T23:00:36	2024-09-18T23:00:47
InstallApplication	acknowledged	2024-09-18T23:00:24	2024-09-18T23:00:36

FileWave does not specify the version when a device honours a request to update an App, it will update to the latest, compatible version currently on the App Store.

Included in FileWave Anywhere, is the option to determine timings of when VPP App upgrades may take place. Please view the following KB on this topic:

https://kb.filewave.com/books/apple-school-business-manager/page/vpp-application-upgrade-timing

Managing Updates

There have been instances where disabling auto updates of VPP Apps has been required; due to unexpected behaviour from the App Store. In such cases, it can be desirable to either block updates completely or block updates per App.

Beyond the above mentioned method to manage VPP App upgrade timings, it may be desirable to block certain versions of an App. FileWave has some additional options for VPP App management.

Overriding this behaviour may be done by adding options within the custom settings:

```
# macOS/Linux

/usr/local/filewave/django/filewave/settings_custom.py
```

Options available are:

Key	Description
SELF_HEAL_APPS_BY_VERSION	Enables/disables all VPP App updates
IGNORE_PREINSTALLED_APPS_SELF_HEAL	Block all update attempts for a defined App by Bundle ID (Unmanaged Apps only)

IGNORE_ITUNES_VERSION	Block updates, not only by a defined App Bundle ID, but only blocks defined version numbers

Directions

Each example below involves editing: settings_custom.py. Any changes require apache to be restarted. Where the App Bundle ID or version is required, this may be observed in a device's Installed App list.

SELF_HEAL_APPS_BY_VERSION

Add the following line will block all updates of all Apps:

```
SELF_HEAL_APPS_BY_VERSION = False
```

To revert this behaviour, either set this as True or remove the entire line.

IGNORE_PREINSTALLED_APPS_SELF_HEAL

This option will only prevent erroneous attempts to update unmanaged Apps, where the device reports an incorrect Bundle ID. As an unmanaged App, it may not be updated by MDM anyway, but installation errors would be seen in the Command History.

Obtain the Bundle ID of the App, then add the following option. For example, to block iMovie, Pages and Keynote:

```
settings.IGNORE_PREINSTALLED_APPS_SELF_HEAL = ("com.apple.iMovie", "com.apple.Pages", "com.apple.Keynote")
```

This is a comma separated list. Add each Bundle ID per App to be blocked for updates.

To revert this behaviour, either remove the Bundle ID no longer required for blocking or remove the entire line.

IGNORE_ITUNES_VERSION

Obtain the Bundle ID and version of the App to be blocked. The settings are set out as:

'Bundle ID': [(version to block, version currently installed)]'Bundle ID': [(version to block, version currently installed)]

```
'Bundle ID': [(version to block, version currently installed)]
```

The below example will:

- Keynote - Block version 3.0 from the iTunes Store if device has version 2.7 installed
- iMovie - Block version 10.1.14 from the iTunes Store if device has version 10.1.13 installed
- Pages - Block version 8.2 from the iTunes Store if device has version 8.2 installed

```
settings.IGNORE_ITUNES_VERSION = {
'com.apple.keynotes': [('3.0', '2.7'),],
'com.apple.iMovie': [('10.1.14', '10.1.13'),],
'com.apple.Pages': [('8.2', '8.2'),],
}
```

The Pages example prevents an App from continually attempting to update, where the version on the iTunes Store matches that on the device, yet the device is still reporting an update is required. Add or remove entries per item to be blocked.

Taking this one step forward, consider the above Keynote example. Device has 2.7 installed, but version 3.0 is set to be ignored. If version 3.1 were to be released, the device would upgrade to this version, if it were the next latest version available on the App Store, after receiving a new InstallApplication command.

To revert the behaviour, either remove the Bundle ID and versions no longer required for blocking or remove this entire code entry.

> ⚠ White Space
> Note, there should be no 'white space' before the added key: spaces, tabs, etc. Doing so will result in the server becoming non-responsive.

Apache should then be restarted:

```
# macOS/Linux

/usr/local/filewave/apache/bin/apachectl graceful
```

Revision #3
Created 12 July 2023 16:09:50 by Andrew Kloosterhuis
Updated 26 September 2024 11:36:32 by Sean Holden