EUD Security Guidance: macOS 10.13+

Information

The Nation Cyber Security Centre has various guides on security. This article covers a section of the End User Device Security Collection and how to use FileWave to monitor or follow the guidance as laid out:

https://www.ncsc.gov.uk/guidance/eud-security-guidance-macos-1013-high-sierra#recommendedpoliciesandsettings

From the document:

> “ This guidance was developed following testing performed on MacBook Pro and MacBook Air devices running macOS 10.13 (High Sierra)
>
> It's important to remember that this guidance has been conceived as a way to satisfy the 12 End User Device Security Principles. As such, it consists of recommendations and should not be seen as a set of mandatory instructions requiring no further thought.
>
> Risk owners and administrators should agree a configuration which balances business requirements, usability and security.

As set out from the list, here are some solutions. For descriptions of Configuration Profiles, please consult FileWave Profile Editor section.

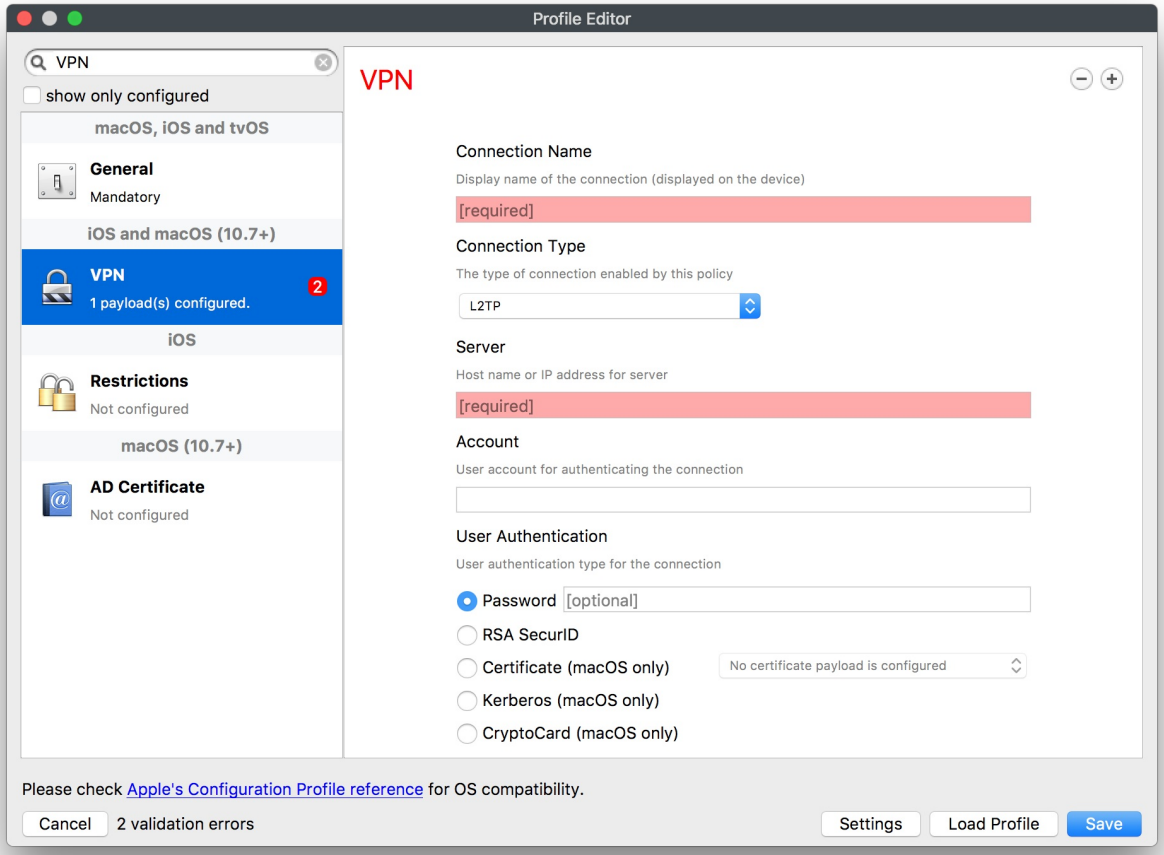
Custom Fields
Below are additionally some custom fields for reporting on the status of pertinent settings. These have been described as if using the new Custom Fields introduced in 12.7+.

For using these Custom Fields with earlier versions of FileWave, these will need to be added as Filesets and edited to write custom values to the client using fwcld.

The Custom Fields provided are examples using English. Where searching for text in response to a command is concerned, if the OS is running a different language, the script may need to be altered to match.

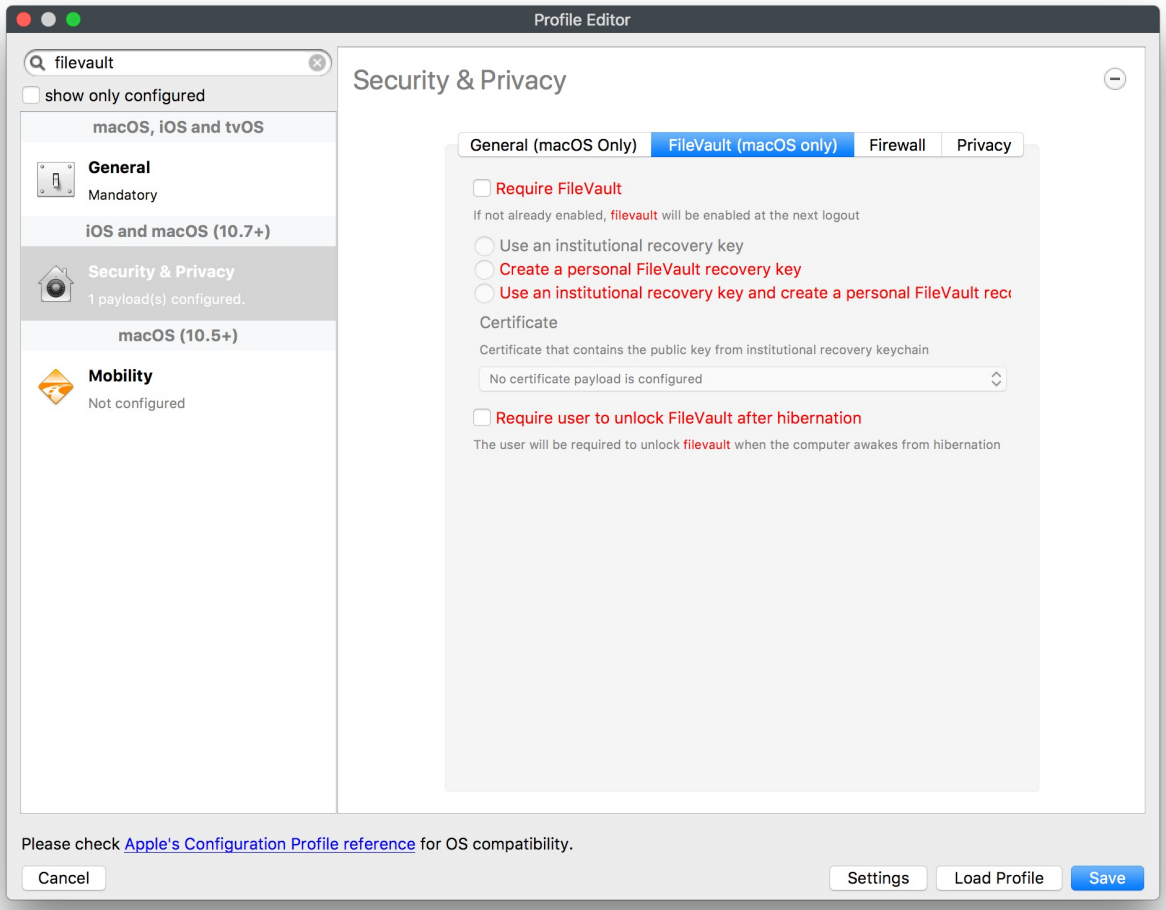
Assured data-in-transit protection, VPN Profile

VPN may be configured using Configuration Profiles



Assured data-at-rest protection, FileVault

FileVault may be configured using Configuration Profiles



Custom Field to show current FileVault status:

```
#! /bin/bash
#
# FileWave client will execute this script. The output will be used as the value of the custom field.
#
# Below is an example of how to read the value of one ENVIRONMENT VARIABLE in your script:

# my_var=${ENV_VAR_NAME}
#

fv_status=$(fdesetup status 2>/dev/null)
error_code=$?

if [ $error_code -ne 0 ]
then
        echo "Error: "$error_code
else
        case $fv_status in

        *"Off"*)
                echo "Off"
                ;;
        *"On"*)
                echo "FileVault"
                ;;
        *)
                echo "$fv_status"
                ;;
        esac
fi

exit 0
```

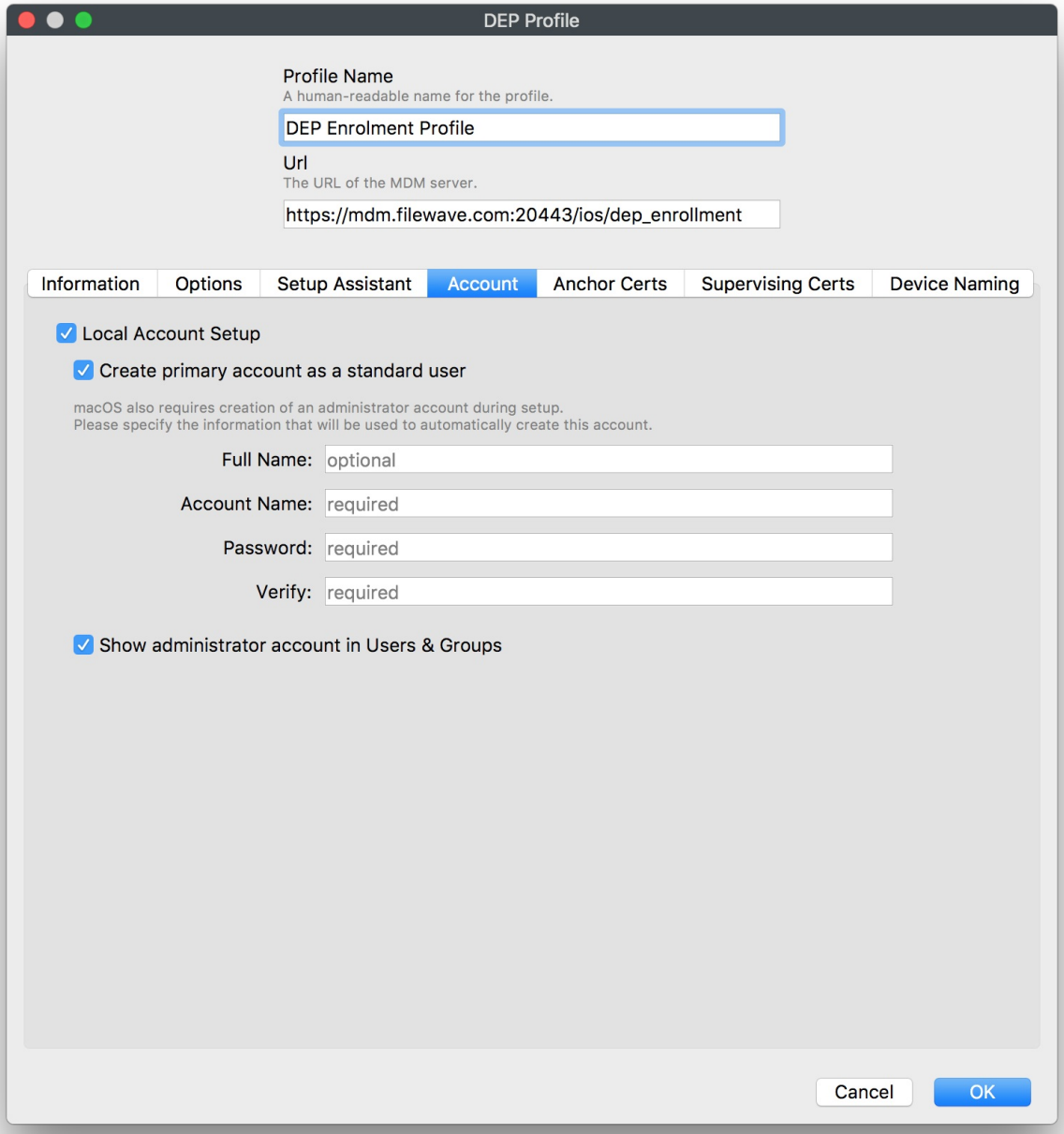
Custom Field to show if device supports FileVault Authenticated Restart. N.B. This requires FileVault to be enabled to work.

```
#! /bin/bash
#
# FileWave client will execute this script. The output will be used as the value of the custom field.
#
# Below is an example of how to read the value of one ENVIRONMENT VARIABLE in your script:

# my_var=${ENV_VAR_NAME}
#

filevault_auth_bypass=$(fdesetup supportsauthrestart) 2>/dev/null

if [ $? -ne 0 ]
then
        echo Unsupported
else
        echo $filevault_auth_bypass  | sed -e 's/t/T/g' -e 's/f/F/g'
fi

exit 0
```

Authentication

> Either:
>
> * Users have two passwords – one for FileVault 2, and one to login and unlock their device (see Provisioning Steps for how to achieve this)
> * Or users have one password which fulfils both requirements.

DEP Profile can include a local admin account to be created during enrolment, which could be hidden.



> “ The user should be required to authenticate to the device in line with your organisation’s authentication policy (see Authentication Policy).
>
> This user’s login password derives a key which encrypts certificates and other credentials, giving access to organisational services.

Secure boot

> “ Set an EFI (firmware) password to make it more difficult for an attacker to modify the boot process. However, with physical access, the boot process can still be compromised.

Firmware password can be set using the provided recipe: Firmware Password (macOS 10.14 Mojave+)

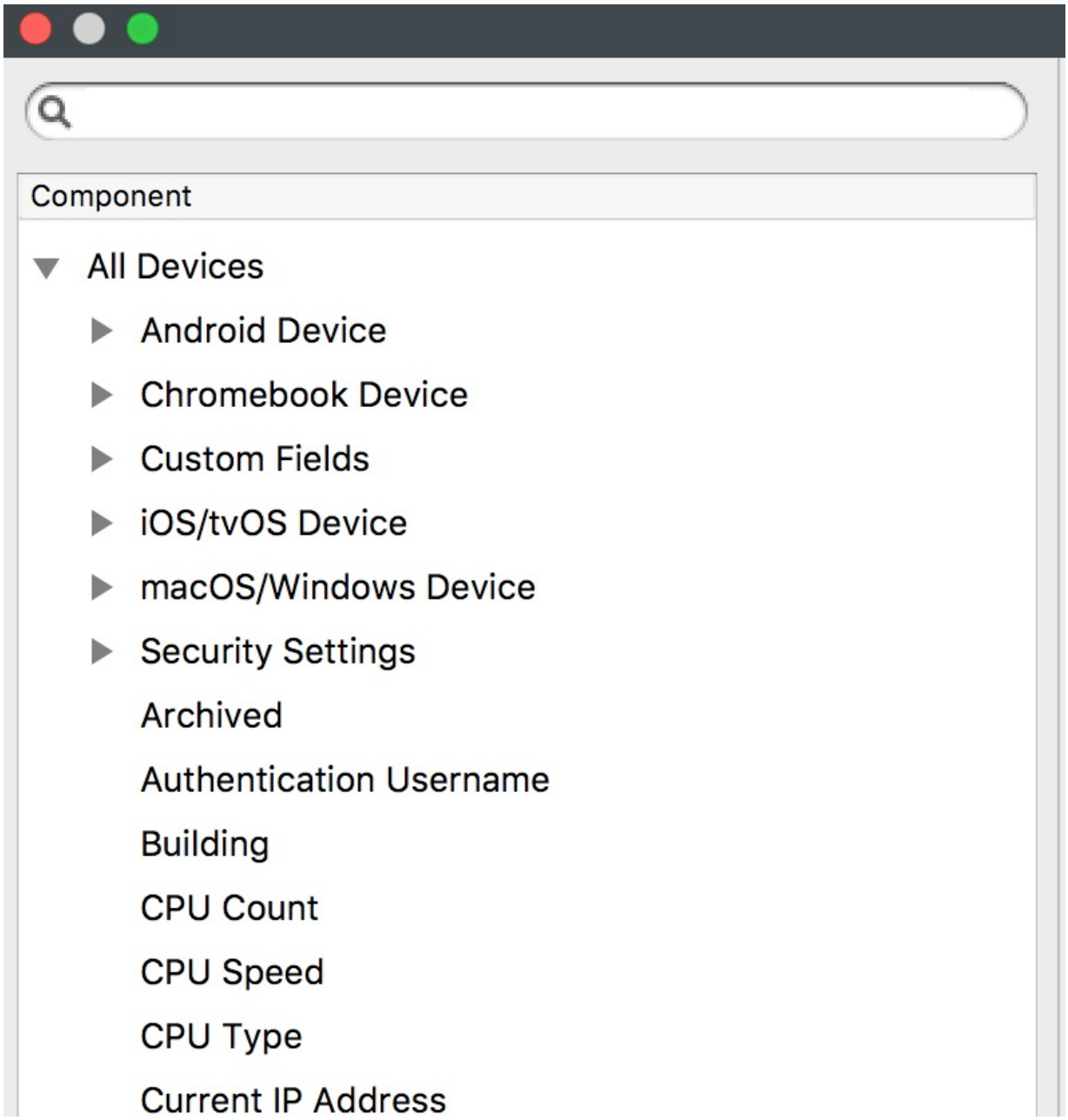
Custom Field to show Firmware Status:

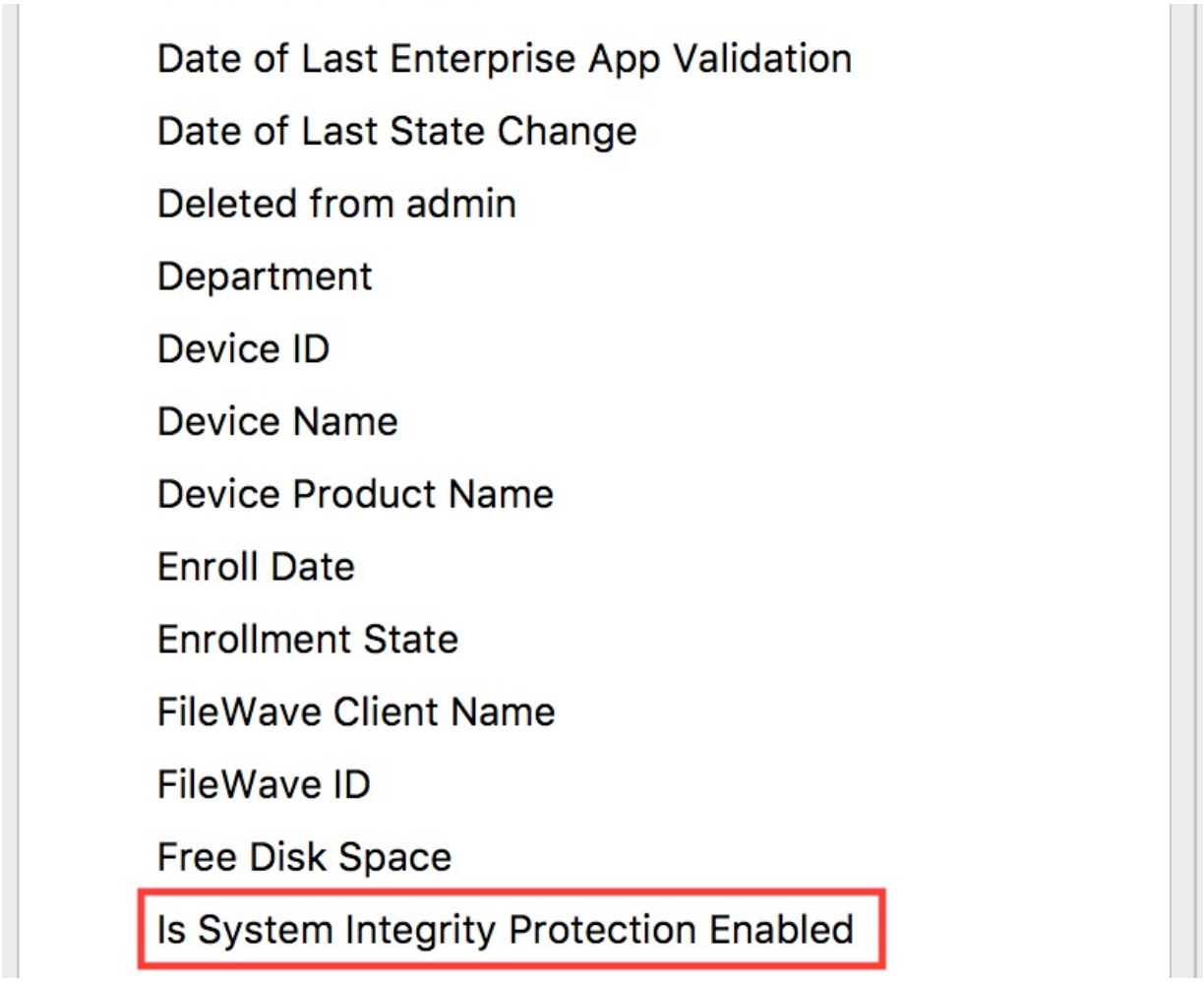
```
#! /bin/bash
#
# FileWave client will execute this script. The output will be used as the value of the custom field.
```

```
#
# Below is an example of how to read the value of one ENVIRONMENT VARIABLE in your script:

# my_var=${ENV_VAR_NAME}
#

if [ -e /usr/sbin/firmwarepasswd ]
then
        firmware_status=$(firmwarepasswd -check | awk '{print $NF}')
        error_code=$?

        if [ $error_code -ne 0 ]
        then
                echo "Error: "$error_code
        else
                echo "$firmware_status"
        fi
else
        echo "N/A"
fi

exit 0
```

Platform integrity and application sandboxing, SIPs

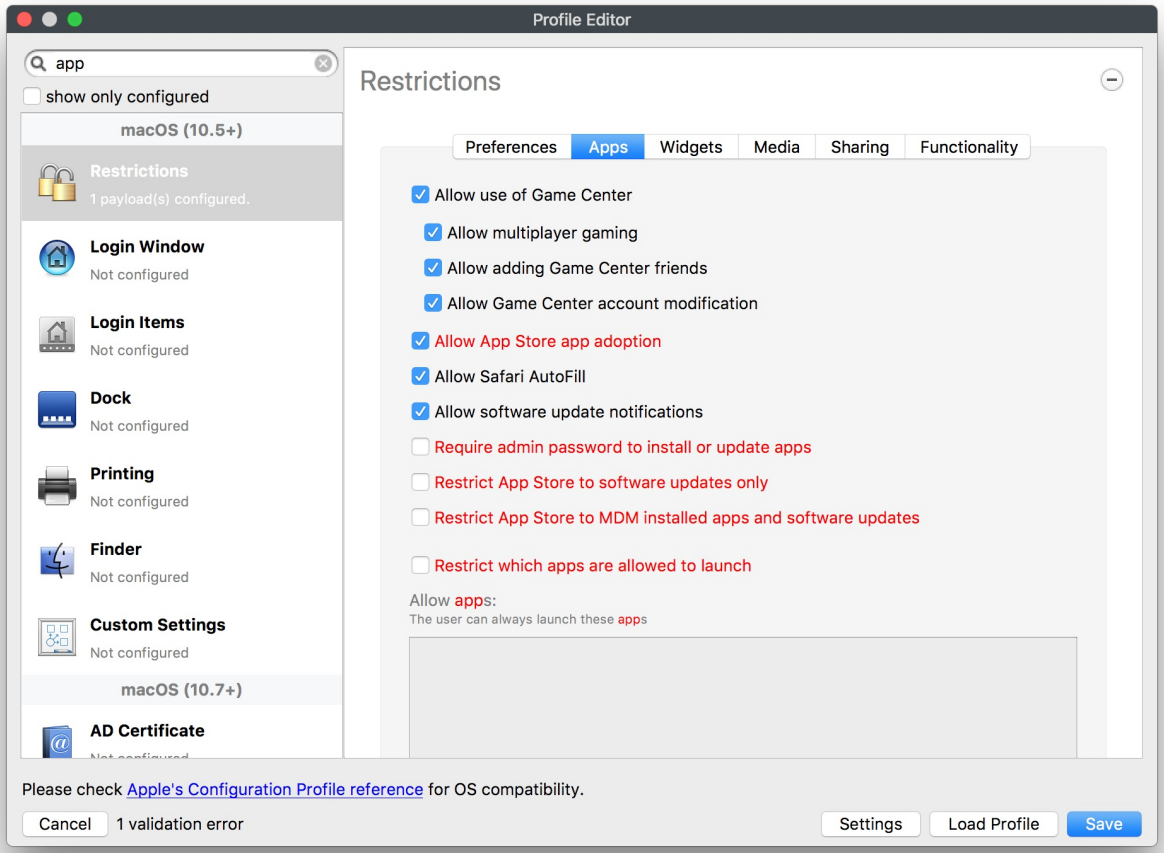
SIPs is on by default and is unlikely to be disabled. However, FileWave has a default inventory item to check that SIP is enabled:

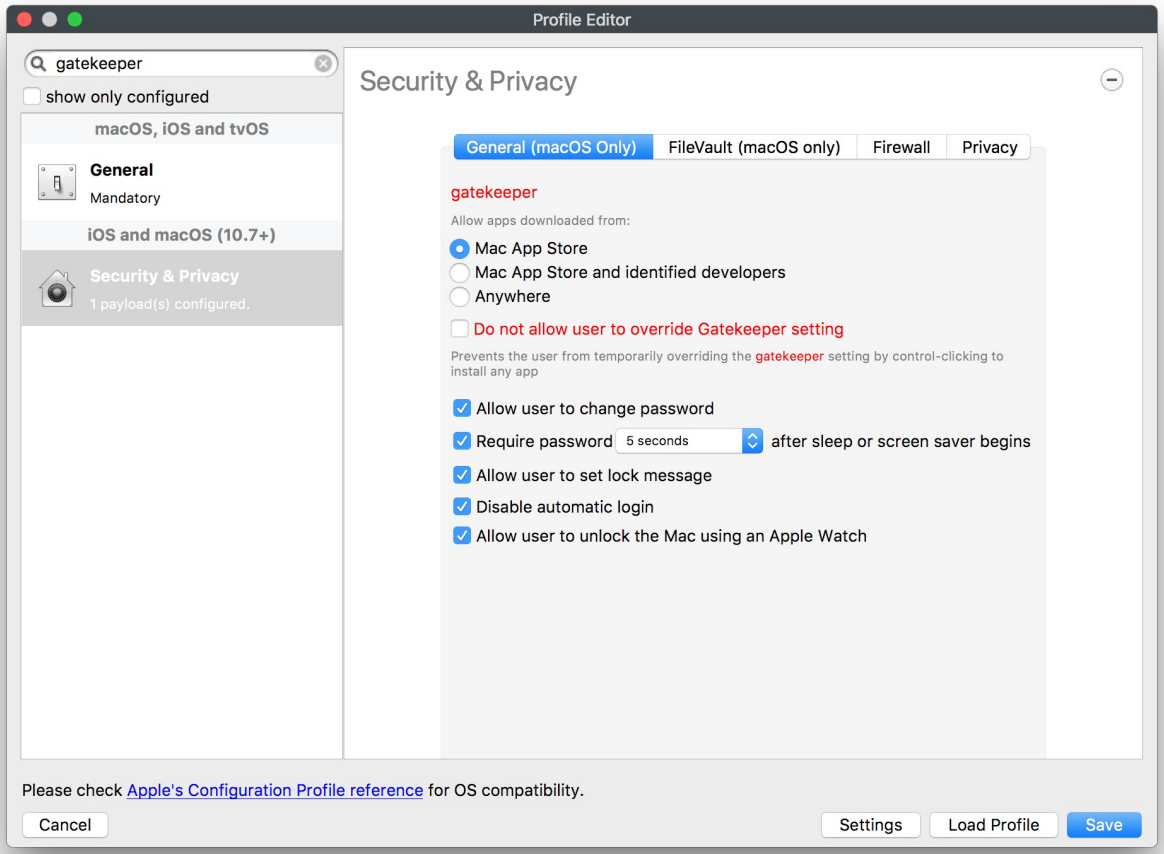




Application whitelisting

> Use the MDM to whitelist default macOS applications. Use GateKeeper to prevent the installation and running of unsigned applications. An organisation application catalogue can also be used which only contains enterprise-approved or in-house applications.





Custom Field to display Gatekeeper version:

```
#! /bin/bash
#
# FileWave client will execute this script. The output will be used as the value of the custom field.
#
# Below is an example of how to read the value of one ENVIRONMENT VARIABLE in your script:


# my_var=${ENV_VAR_NAME}
#

defaults read /var/db/gkopaque.bundle/Contents/Info.plist CFBundleShortVersionString

exit 0
```

Malicious code detection and prevention

> XProtect is built into macOS. It has a limited signature set which is maintained by Apple to detect widespread malware. XProtect will also restrict vulnerable plugin versions (such as Java) to limit exposure. Several third-party anti-malware products also exist which attempt to detect malicious code for this platform. Content-based attacks can be filtered by scanning capabilities in the organisation.

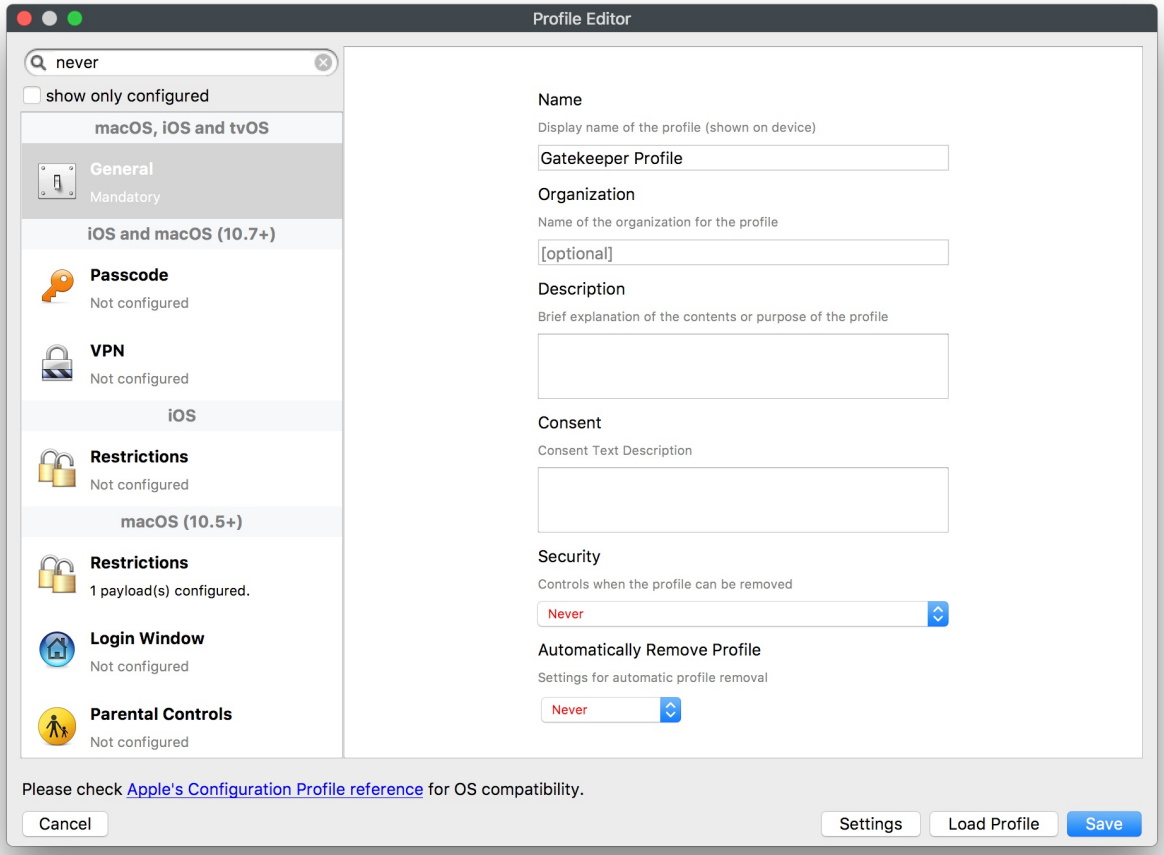
Custom Filed to display XProtect version:

```
#! /bin/bash
#
# FileWave client will execute this script. The output will be used as the value of the custom field.
#
# Below is an example of how to read the value of one ENVIRONMENT VARIABLE in your script:


# my_var=${ENV_VAR_NAME}
#

defaults read /System/Library/CoreServices/CoreTypes.bundle/Contents/Resources/XProtect.meta.plist Version

exit 0
```

Security policy enforcement

> Mark MDM profiles as non-removable so the user cannot remove them and alter their configuration.

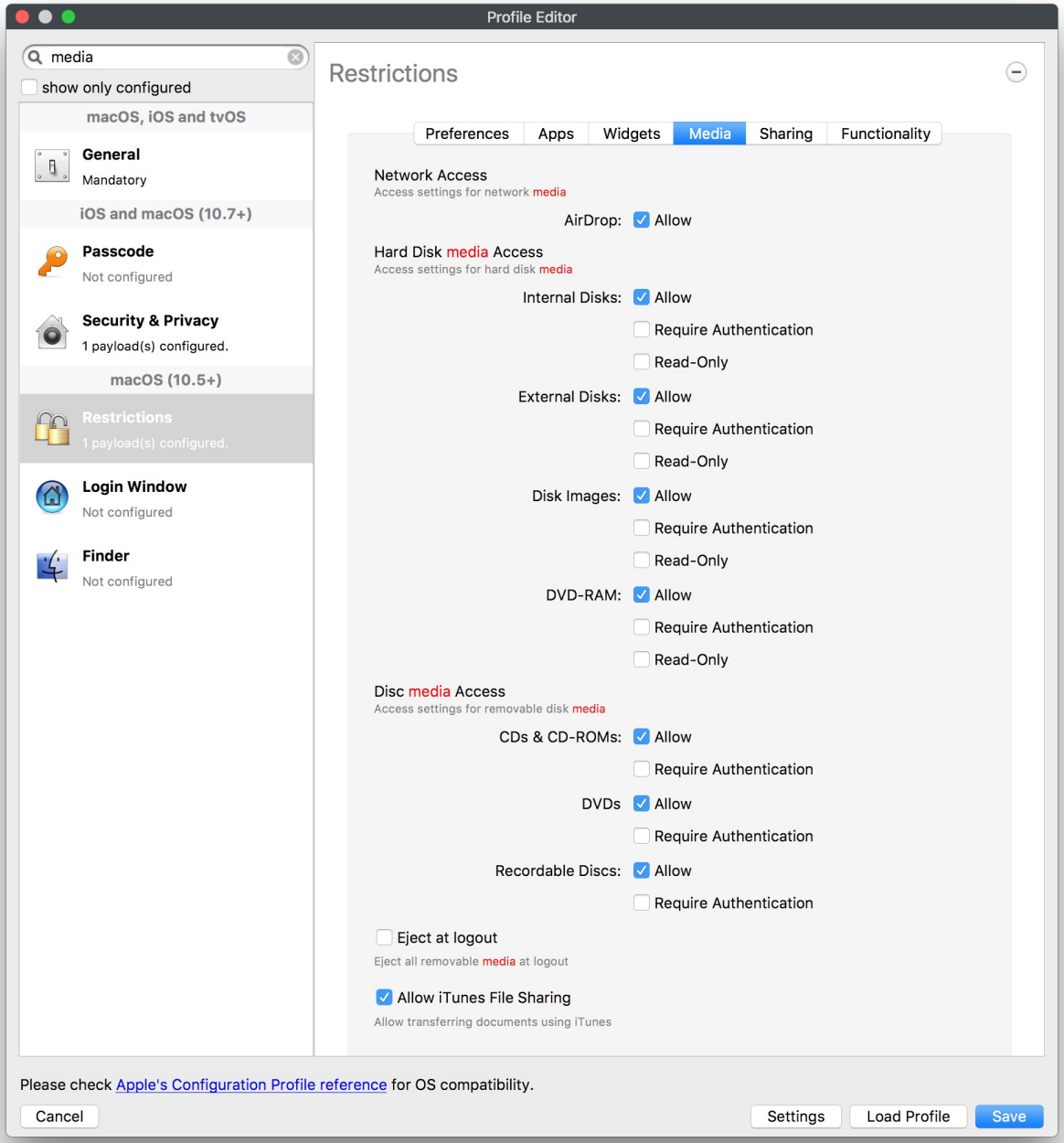
General setting of each profile has an option to deny profile removal under the Security option.



External interface protection

> “ USB removable media can be blocked through MDM if required. If an EFI password is set, DMA is only possible when the device is booted and unlocked.

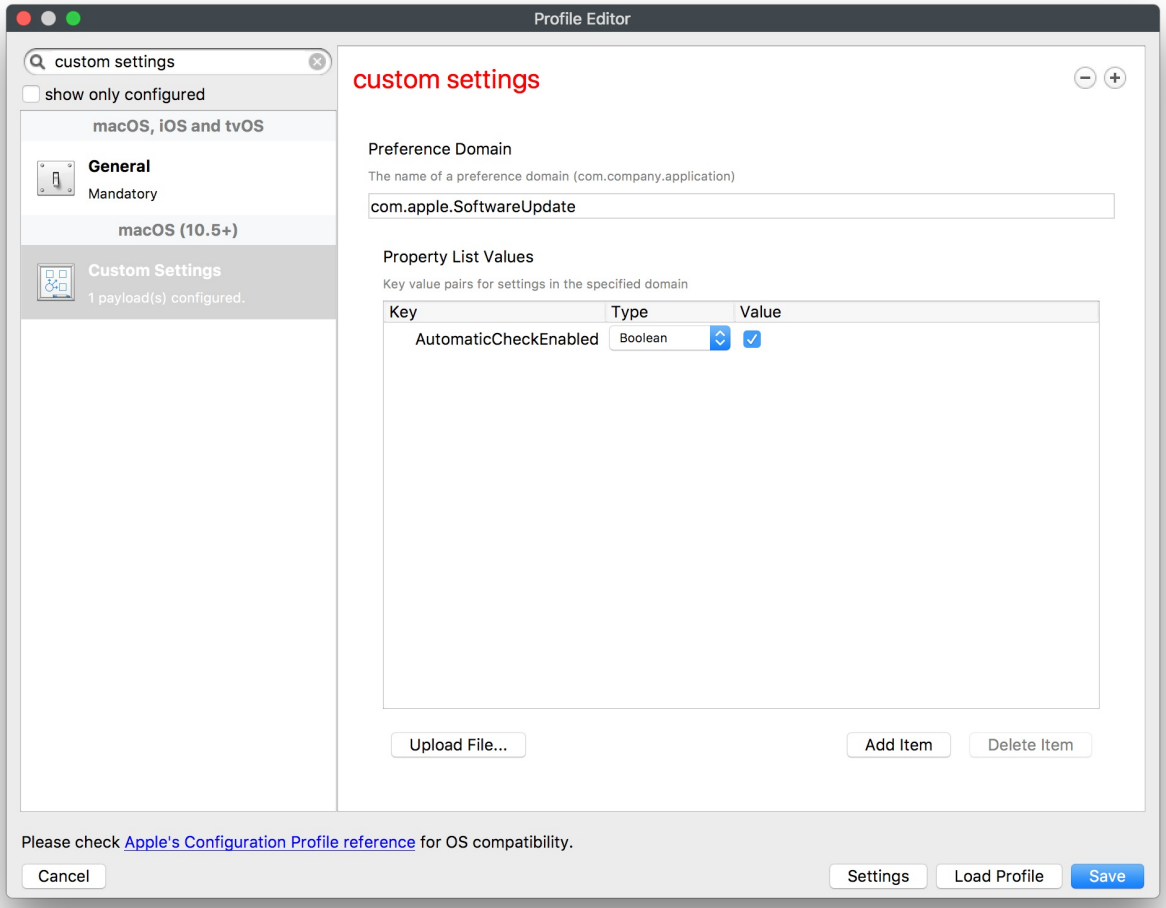
Restrictions profile may be used to control access to external media



Device update policy

> “ MDM can be used to audit which App Store software and OS versions are installed on a device. The attached script will turn on automatic updates, but this cannot be achieved remotely with MDM.

By default, this cannot be achieved over MDM. However, FileWave has the option for a Custom Settings profile that could control this over MDM



Event collection for enterprise analysis

> “ macOS logs can be viewed by a local administrator on device, or from a distance using remote administration tools. Third-party software can also be used to automate log collection.

Client Info has an option to view Command History. Client Info and Client Monitor have the option to view FileWave logs by default.

Viewing macOS logs would involve another method. Options could include:

- Fileset Activation Script to read the file. Output of scripts can be viewed in the Fileset Status view; right click on Script and choose 'View Script Output'.
- Remote Control of device
- Fileset Script to upload these files to a secure share

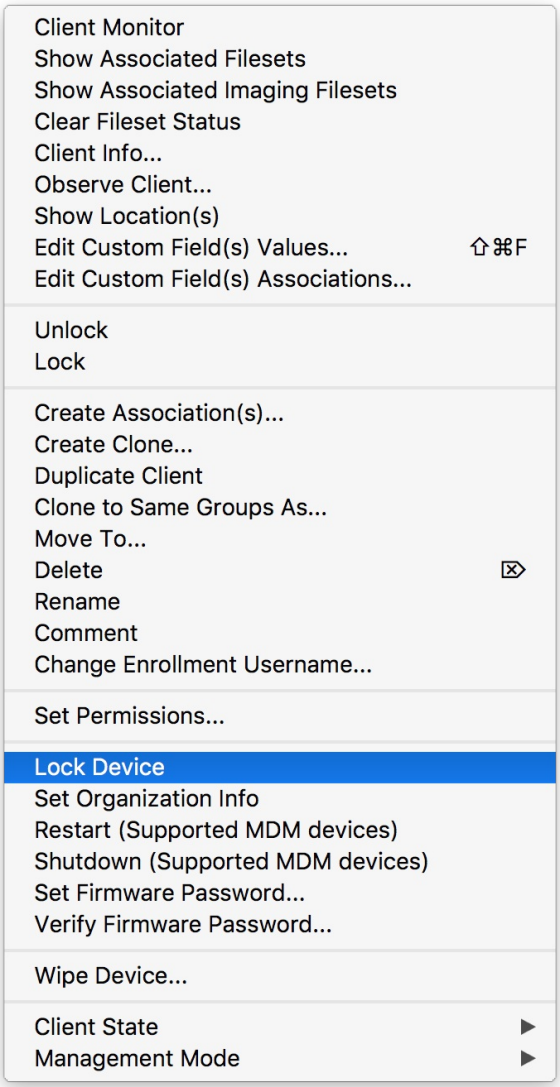
Example Activation Script:

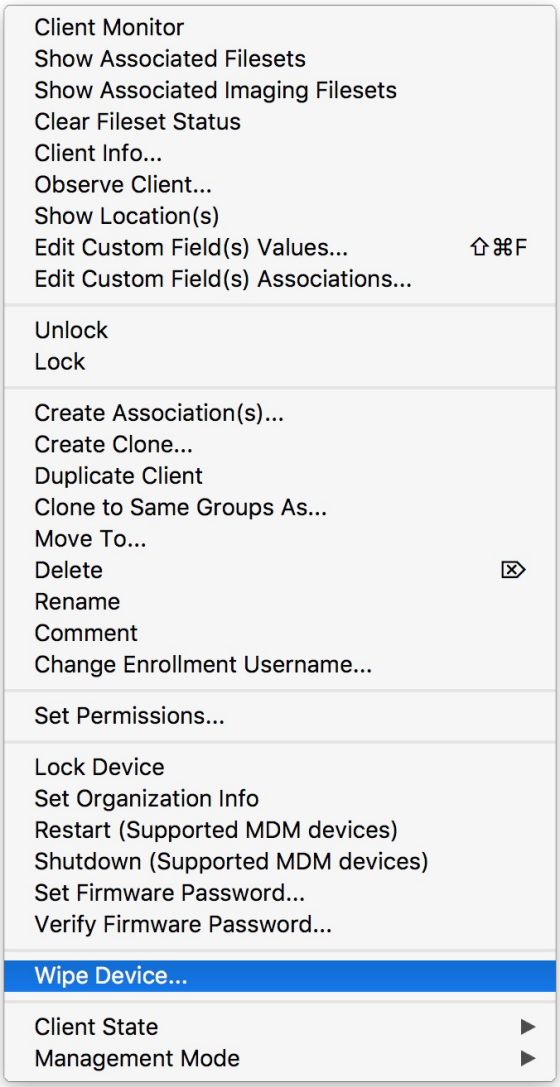
```
#!/bin/bash
cat /private/var/log/system.log

exit 0
```

Incident response

“macOS devices can be locked, wiped, and configured remotely by their MDM.”

Device context menu has options to either Lock or Wipe a device. FileWave has extensive options for remotely configuring MDM enrolled devices.


Client Monitor
Show Associated Filesets
Show Associated Imaging Filesets
Clear Fileset Status
Client Info...
Observe Client...
Show Location(s)
Edit Custom Field(s) Values... ⇧⌘F
Edit Custom Field(s) Associations...
Unlock
Lock
Create Association(s)...
Create Clone...
Duplicate Client
Clone to Same Groups As...
Move To...
Delete
Rename
Comment
Change Enrollment Username...
Set Permissions...
Lock Device
Set Organization Info
Restart (Supported MDM devices)
Shutdown (Supported MDM devices)
Set Firmware Password...
Verify Firmware Password...
Wipe Device...
Client State
Management Mode



Additional Custom Fields

Firewall Status:

```
#! /bin/bash
#
# FileWave client will execute this script. The output will be used as the value of the custom field.
#
# Below is an example of how to read the value of one ENVIRONMENT VARIABLE in your script:

# my_var=${ENV_VAR_NAME}
#

fw_status=$(/usr/libexec/ApplicationFirewall/socketfilterfw --getglobalstate 2>/dev/null)
error_code=$?

if [ $error_code -ne 0 ]
then
        echo "Error: "$error_code
else
        case $fw_status in

        *"disabled"*)
                echo "Off"
                ;;
        *"enabled"*)
                echo "Enabled"
                ;;
        *)
                echo "$fw_status"
```

```
                ;;
        esac
fi

exit 0
```

Local Admins:

```
#! /bin/bash
#
# FileWave client will execute this script. The output will be used as the value of the custom field.
#
# Below is an example of how to read the value of one ENVIRONMENT VARIABLE in your script:

# my_var=${ENV_VAR_NAME}
#

/usr/bin/dscl . -read /Groups/admin GroupMembership | cut -d " " -f 3- | tr " " ","

exit 0
```

Hidden Admins:

```
#! /bin/bash
#
# FileWave client will execute this script. The output will be used as the value of the custom field.
#
# Below is an example of how to read the value of one ENVIRONMENT VARIABLE in your script:

# my_var=${ENV_VAR_NAME}
#
declare -a HiddenAdmins=()

function check_hidden {

        while [ $# -gt 0 ]
        do
                IsHidden=$(dscl . read /Users/$1 IsHidden 2>/dev/null | awk '{print $NF}')

                if [[ $IsHidden -eq 1 ]]
                then
                        HiddenAdmins+=($1)
                fi

                shift
        done
}

check_hidden $(/usr/bin/dscl . -read /Groups/admin GroupMembership | cut -d " " -f 3-)

echo ${HiddenAdmins[@]} | tr " " ","

exit 0
```

Visible Admins:

```
#! /bin/bash
#
# FileWave client will execute this script. The output will be used as the value of the custom field.
#
# Below is an example of how to read the value of one ENVIRONMENT VARIABLE in your script:

# my_var=${ENV_VAR_NAME}
#

#!/bin/bash

declare -a VisibleAdmins=()
```

```
function check_hidden {

        while [ $# -gt 0 ]
        do
                IsHidden=$(dscl . read /Users/$1 IsHidden 2>/dev/null | awk '{print $NF}')

                if [[ ! $IsHidden -eq 1 ]]
                then
                        VisibleAdmins+=($1)
                fi

                shift
        done
}

check_hidden $(/usr/bin/dscl . -read /Groups/admin GroupMembership | cut -d " " -f 3-)

echo ${VisibleAdmins[@]} | tr " " ","

exit 0
```

Installed Security Updates:

```
#!/bin/zsh

declare -a output_array=()
get_pkg_name=0

function check_duplicate {

        if [ ${#output_array[@]} -eq 0 ]
        then
                output_array+=($1)
        else
                array_counter=0

                while [ $array_counter -lt ${#output_array[@]} ]
                do
                        array_counter=$(( array_counter + 1 ))

                        if [[ "$1" == "${output_array[${array_counter}]}" ]]
                        then
                                break
                        fi

                        if [ $array_counter -eq ${#output_array[@]} ]
                        then
                                output_array+=($1)
                        fi
                done
        fi
}

/usr/libexec/PlistBuddy -c 'Print ' /Library/Receipts/InstallHistory.plist | while read line
do
        case "$line" in

                *"Security Update")
                        get_pkg_name=1
                ;;
                *"Security Update"*)
                        check_duplicate "${line##* }"
                ;;
                *"com.apple.pkg.update.os"*|*"com.apple.pkg.update.10"*)
                        if [ $get_pkg_name -eq 1 ]
                        then
                                sec_number=$( echo "$line" | awk -F "." '{print $(NF-1)}' | sed 's/[A-Za-z]*//g')
                                check_duplicate "${sec_number}"
                                get_pkg_name=0
                        fi
```

```
                ;;
        esac
done

echo ","${output_array}"," | tr " " ","

exit 0
```

Firmware Unlock Seed (password recovery key). A new key is generated with each password change. Firmware password changes require a reboot before reporting correctly. Key may only be used by Apple.

```
#! /bin/bash
#
# FileWave client will execute this script. The output will be used as the value of the custom field.
#
# Below is an example of how to read the value of one ENVIRONMENT VARIABLE in your script:

# my_var=${ENV_VAR_NAME}
#

# Retrieve firmwarepassword unlockseed.  Response may only be used by Apple to unlock the device.

# Firmware status
check_firmware=$(firmwarepasswd -check | awk '{print $NF}')

if [[ "$check_firmware" != "No" ]]
then
        unlock_seed=$(firmwarepasswd -unlockseed &>/dev/null)
        if [ $? -eq 0 ]
        then
                echo $unlock_seed
        else
                echo Error
        fi
else
        echo Disabled
fi

exit 0
```

Preparation for deployment

> “ The steps below should be followed to prepare your organisation's infrastructure for hosting a deployment of these devices:
>
> Set up an MDM server (e.g. Profile Manager on macOS Server). This may require setting up the Open Directory component of a macOS Server.
>
> Ensure all Configuration Profiles are signed to prevent modification in transit, or post install
>
> Create policies on Profile Manager for:
> * Ensure 'Use SSL' is selected for all server settings
> * VPN
> * Passcode
> * Disk encryption and key escrow
> * Exchange/Mail/Calendar Settings.
> * Disabling access to the Preference Panes in Restrictions (macOS) for iCloud and Network as access to these could be used to disable the VPN.”

FileWave can be used to mimic Profile Manager and much more.

It is worth noting however, that FileWave does not have the option to sign Configuration Profiles. This is a current Feature Request: FW-19687.

Recommendation in this case would be to use Profile Manager to sign the profile and then import this into FileWave. It is also possible to achieve this from the macOS command line:

```
/usr/bin/security cms -S -N "Certificate Common Name" -I "/path/to/original.mobileconfig" -o
"/path/to/create/signed.mobileconfig"
```

Signing certificates can be generated if you have a Apple developer account or if you already have a macOS Server running and then added to the keychain prior to signing.

Policy Creations

- YES - Use SSL in various Payloads
- YES - VPN payloads
- YES - Passcode payloads
- YES - Disk Encryption with Personal and Institutional Recovery Key
- Partly - Profile may be applied to enforce both/either Institutional and Personal Recover Key. Currently storing the Personal Recovery Key is not supported. There is a Feature Request for this: FW-20543
- YES - Exchange/Mail/Calendar Settings.
- YES - Disabling access to the Preference Panes in Restrictions (macOS) for iCloud and Network as access to these could be used to disable the VPN.

> “ “Additional Consideration
>
> * Whitelist applications to further reduce the risk of malicious code being execute
> * Tighten permissions on USB mass storage and optical devices to help prevent data loss through removable media
> * Use Restrictions to blacklist locations from which users should not run applications, or whitelist trusted applications that users are allowed to run
> * Include internal CA Certificates where appropriate to ensure users can authenticate network services
> * Include corporate network profiles (e.g. 802.1X or Wi-Fi) to ensure that network access credentials are distributed securely”

- YES - Whitelist applications to further reduce the risk of malicious code being execute
- YES - Tighten permissions on USB mass storage and optical devices to help prevent data loss through removable media
- YES - Use Restrictions to blacklist locations from which users should not run applications, or whitelist trusted applications that users are allowed to run
- YES - Include internal CA Certificates where appropriate to ensure users can authenticate network services
- YES - Include corporate network profiles (e.g. 802.1X or Wi-Fi) to ensure that network access credentials are distributed securely

> Feature Requests
> Where Feature Requests have been noted, please view release notes for added features in future updates.

Revision #3
★Created 16 July 2023 15:42:53 by Josh Levitsky
Updated 4 November 2024 13:54:09 by Josh Levitsky