Munki Software Management Tool for Macs, Whitepaper

Prepared by Allister Banks, September, 2015

Background

This whitepaper explains the function Munki performs, and some current implementation details. With ongoing input from Facebook, Dropbox, and Google, the Munki open source software management tool (created and maintained by Greg Neagle at Walt Disney Animation Studios) streamlines administration and provide customers with benefits that encourage usage. Nothing surpasses it for platform-specific, Mac-native software interaction.

Executive Summary

Rapid and programmatic installation, update, removal, and assignment of software sets Munki apart from other tools that purport to deliver software to Macs. It comprises of a client app that can enable standard users to perform installs or updates, work around poorly-packaged software like Adobe products, and most everything except Mac App Store software can be maintained through it. This allows customers ‘one-stop-shopping’ and a go-to app instead of each product or vendor bothering them individually. It can set a countdown date to forcibly log a machine off if installation is required, and is built around a model that employs controls to test software in the ‘one-some-many’ model. Uninstallation's can remove unused or licensed software that should be reassigned. Server-side, it just requires a set of folders on any generic web server, and can be secured with per-computer certificates to allow or revoke access granularly.

Explanation of Features and Concepts

Munki has a rich ecosystem of tools for all aspects of admin interaction, and it ships with command-line utilities that can automate various tasks. The administrator can remotely make configuration changes, check the flat files that make up those settings into version control, and deploy them in a staged rollout to get feedback as needed. The most fundamental concepts to understand are its customer-facing application Managed Software Center, and the files stored server-side, like manifests, Catalogs, and Pkginfo files.

Managed Software Center

As pictured on the cover, the Managed Software Center application provides a software store-like portal to offer optional software and provide self-service uninstalls as well. Pending actions like installs, updates, or removals that are enforced by the administrator would appear on the top-right ‘Updates’ tab, Self-service type interactions are on the first Software tab, and there are options to customize the sidebar links, images to brand it for the organization, and other links like in the footer or Help menu.

Manifests

One way to put it is a manifest is a list of the software-related configurations you’d like to specify for one or many computers, which may not even need to be displayed in Managed Software Center. Instead of going into many of the specifics regarding it, the most important things to know are that this is where you take the inventory of what Munki can interact with and specify how it will carry out its designated tasks. It can ‘manage’ the installation, update, or removal of software or configuration profiles, or allow software to be optionally installed.

It needs to include a ‘catalog’, which is how you can specify how ‘bleeding-edge’ a selection of the software is that you’d like to deliver, as we’ll cover in a moment. Manifests have the ability to be

specified per-workstation, and then include others, so each individual machine can inherit from a standard set, and the majority of configurations only need to be set in one place. Often people include manifests named for a department or function.

Catalogs and Pkginfo Files

There is the risk of instability or undesired behavior when pushing out software as soon as it is released from a vendor. To group software into versioned ‘branches’ or ‘tracks’ that refer to how recently a piece of software was released, catalogs exist to specify how much risk is acceptable for the manifest that will consult it for software. A ‘waterfall’ or ‘cooling-off’ process of releasing the software can be followed by first importing a new release of software into the ‘unstable’ catalog, then ‘promoting’ it to the ‘testing’ catalog after a pre-determined time period, and finally the ‘production’ catalog.

Software is commonly distributed inside of an archive of some kind, to be dragged-and-dropped into the root Applications folder. This requires administrator rights if the software is not from the Mac App Store, which would in turn require an Apple ID. Munki runs as root and can therefore enable standard users to choose the best time for them to run updates, and get updates from many different sources taken care of all at once.

Similar to MSI format on Windows, Macs install files that end in ‘.pkg’. Munki’s ‘Pkginfo’ files track the metadata about a software payload or release, and can be helpful when correcting or otherwise ensuring necessary actions are performed after installation or for an uninstallation. They can also create relationships between software, so that updates are automatically recognized without further specification, and uninstallation likewise can track down related software to remove. This means less taking packages apart & rebuilding them every time a new update comes out.

Enforcement and Deployment

A common concern for administrators is the lack of uptake on critical updates, which Munki can assist with through the use of the Force Install After Date pkginfo key[1]. After a countdown, users can be forcefully logged out of the computer so that an installation session can proceed. This is used very infrequently, since many updates can be applied via another pkginfo option, ‘unattended install’. While you would think it could cause instability to update software while it is running, Flash for example has enough issues resulting in frequent critical updates that it is a common practice to use the unattended install option instead of constantly interrupting users.

Since Munki runs and checks into a server continuously and uses information about its current state anyway, there are various products that piggy-back on its client runs to send inventory information so you can tell exact install coverage. With that inventory format, folks have even detected outdated firmware versions, as detailed in a post on the AFP548 community blog[2].

---

[1] https://github.com/munki/munki/wiki/Pkginfo-Files#force-install-after-date

[2] https://www.afp548.com/2015/03/05/thunderstrike-need-to-know/

PCI DSS

Having these methods of enforcing a patch and checking its application at the same time, Munki goes hand-in-hand with the PCI-DSS requirement 6.2b to apply critical patches within one to three months of release[3].

NIST 800-53

Ensuring users can run with standard privileges and only install reviewed and approved software from a central repository ensures compliance with NIST CM-7, and CM-11 baselines. CM-8[4] in particular calls out the flexibility of how Munki's model allows customizations if a small group of hosts need a particular configuration, and CM-10 addresses the major advantage of open source software: you can review the code yourself, and as Munki is Apache licensed there are no issues about patents or copyright-related issues.

As Munki configurations are stored as flat files that can be put into version control, keeping the previous state of a deployment (as per CM-2(3)[5]) is a part of the workflow that can be rolled back to in case of issues, although disallowing rollbacks is another feature.

HIPAA and Secure Communication

As specified in the HIPAA Administrative Simplification 164.312(a)[6], Munki's inventory runs can be leveraged as an event around which to send collected log data to a central point for analysis. Commonly the simple web server that Munki needs can be configured to only grant access over https with client certificates per-device through the use of a Certificate Authority. This allows revocation of a client that goes missing, denying unauthorized access of resources.

Conclusion

Munki has the advantage of a rapid yet stable release cycle, which makes its software ready for use months before new OS versions are released, and many workarounds to combat the instability introduced by vendors who don't understand the Mac platform well. Customers get better notifications, one-stop shopping and application of updates, while admins get secure and optimized management and maintenance experience.

---

[3] https://www.pcisecuritystandards.org/documents/PCI_DSS_v3-1.pdf pg. 53

[4] http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r4.pdf pg. F-73

[5] http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r4.pdf pg. F-65

[6] http://www.hhs.gov/ocr/privacy/hipaa/administrative/combined/hipaa-simplification-201303.pdf pg. 67