

Apex Support Bulletin: Deploying a MAU Caching Server

Revision 1.1 [October 21, 2016]
Contact pbowden@microsoft.com

Summary

Microsoft AutoUpdate (MAU) is a utility that detects, downloads and applies updates to Microsoft applications installed on macOS. Specifically, MAU supports Office 2016, Office 2011, Skype for Business and Lync apps. MAU is not used for Microsoft apps that are downloaded from the Mac AppStore.

By default, MAU will perform version checks against Microsoft’s Content Delivery Network (CDN) on the Internet to determine whether the locally installed app has an update available. If an update is available, MAU will determine the smallest package to download to bring the locally installed version of the app up-to-date.

There are two scenarios where enterprise IT admins might want better control over the update workflow:

1. The ability for MAU to use a local network source for retrieving update packages instead of the Microsoft CDN on the Internet. This scenario is good for ‘branch’ scenarios, and cases where Internet bandwidth is limited. For this scenario, MAU can be configured to use an **‘UpdateCache’**.
2. An enterprise might want to have strict control on which version of Office applications can be installed. For example, Microsoft releases production quality updates on the second or third Tuesday of each month. An enterprise might want to temporarily prevent users from updating to the new build to verify compatibility with custom applications. For this scenario, MAU can be configured to use a custom **‘ManifestServer’**.

Both of the scenarios above can be deployed independently, or together, depending on the requirements of the business. MAU 3.8 or later is required to support both of these scenarios.

How MAU Works

MAU detects application updates every 12 hours by checking a version number embedded in an XML file (known as a ‘manifest’) on the Internet and comparing that against the version of the locally installed app. If the background daemon notices that the XML file references a newer version than what is installed, the full MAU application window is opened and users are prompted to update.

The exact URL of the XML file is dictated by two factors 1) the update channel that the user is subscribed to 2) the identifier of the application that is being checked. The following end-points are the base URLs for each of the channels that MAU supports:

ChannelName	Channel Purpose	Base URL
Production	Highest-quality monthly releases	https://officecdn.microsoft.com/pr/C1297A47-86C4-4C1F-97FA-950631F94777/OfficeMac/
External	Insider Slow – High quality, early access builds	https://officecdn.microsoft.com/pr/1ac37578-5a24-40fb-892e-b89d85b6dfaa/OfficeMac/
InsiderFast	Insider Fast – Good quality weekly builds	https://officecdn.microsoft.com/pr/4B2D7701-0A4F-49C8-B4CB-0C2D4043F51F/OfficeMac/

Application identifiers consist of a 4-character language identifier, 4-character app name and 2-character version number. Typical examples are as follows:

Application	Full Application Identifier
Word 2016 for Mac	0409MSWD15
Excel 2016 for Mac	0409XCEL15
PowerPoint 2016 for Mac	0409PPT315
Outlook 2016 for Mac	0409OPIM15
OneNote 2016 for Mac	0409ONMC15
Office 2011 for Mac (English)	0409MSOf14
Lync 2011 for Mac	0409UCCP14
Skype for Business Mac	0409MSFB16
Microsoft AutoUpdate	0409MSau03

Important notes:

1. The internal version of Office 2016 for Mac applications is ‘15’, whereas the internal version of 2011 applications is ‘14’
2. Office 2016 applications are language-neutral builds and use a fixed language identifier of 0409, regardless of the language preference
3. Office 2011 updates are delivered as a suite, so all 2011 apps use a single app name of ‘MSOf’

If you know the channel and application, you can determine the exact URL of the XML file that MAU will use to check for updates. For example, PowerPoint 2016 on the Insider Fast channel will use <https://officecdn.microsoft.com/pr/4B2D7701-0A4F-49C8-B4CB-0C2D4043F51F/OfficeMac/0409PPT315.xml> to check for updates.

In addition to the XML file, a Microsoft-signed security catalog (.CAT) of the same name is used to verify that both the XML file hasn’t been tampered with, and the update packages have not been altered in any way. The combination of XML and CAT are commonly known as ‘collateral’. MAU requires both the XML and CAT file to be present to successfully detect updates.

Scenario 1: Deploying an ‘UpdateCache’ Service

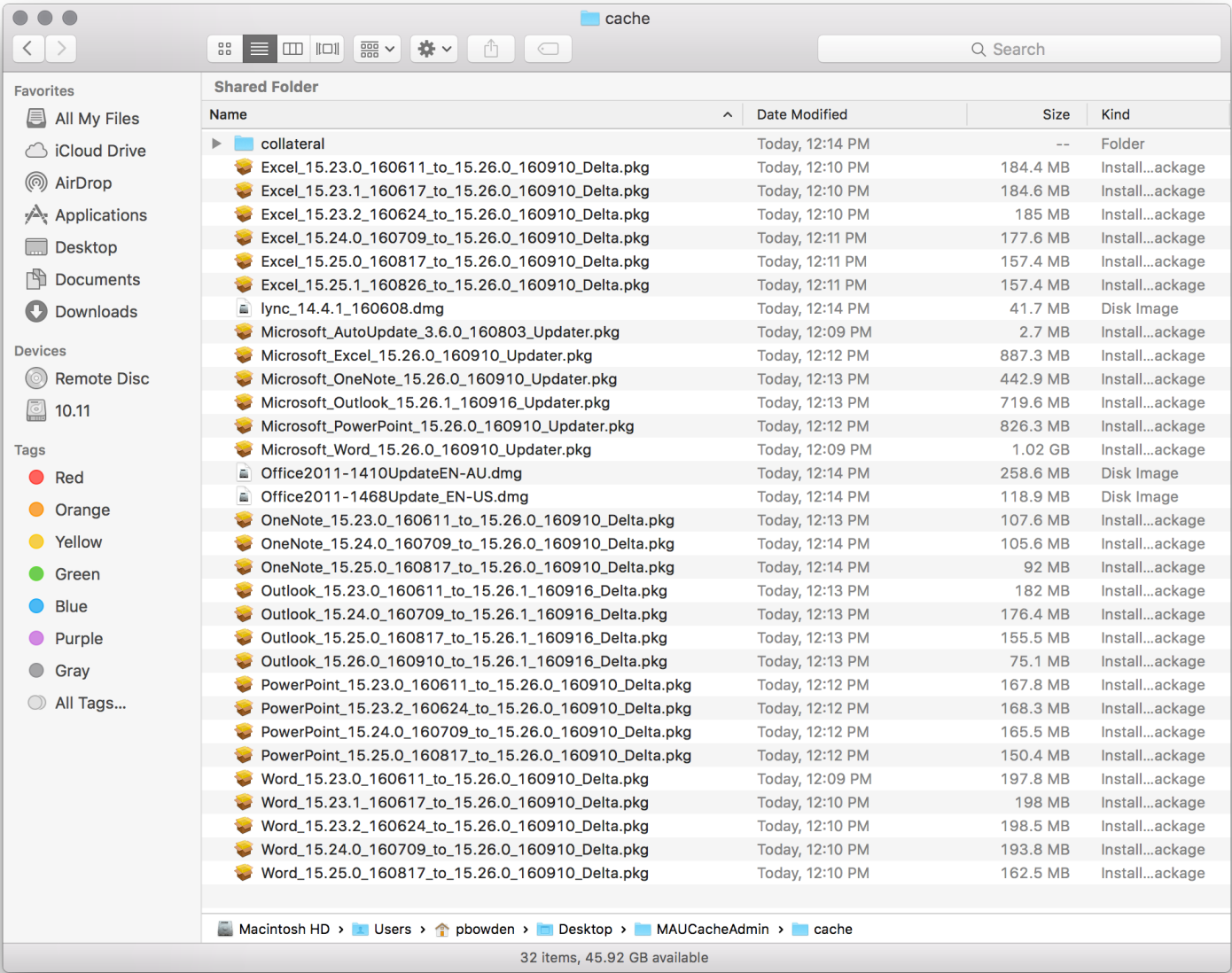
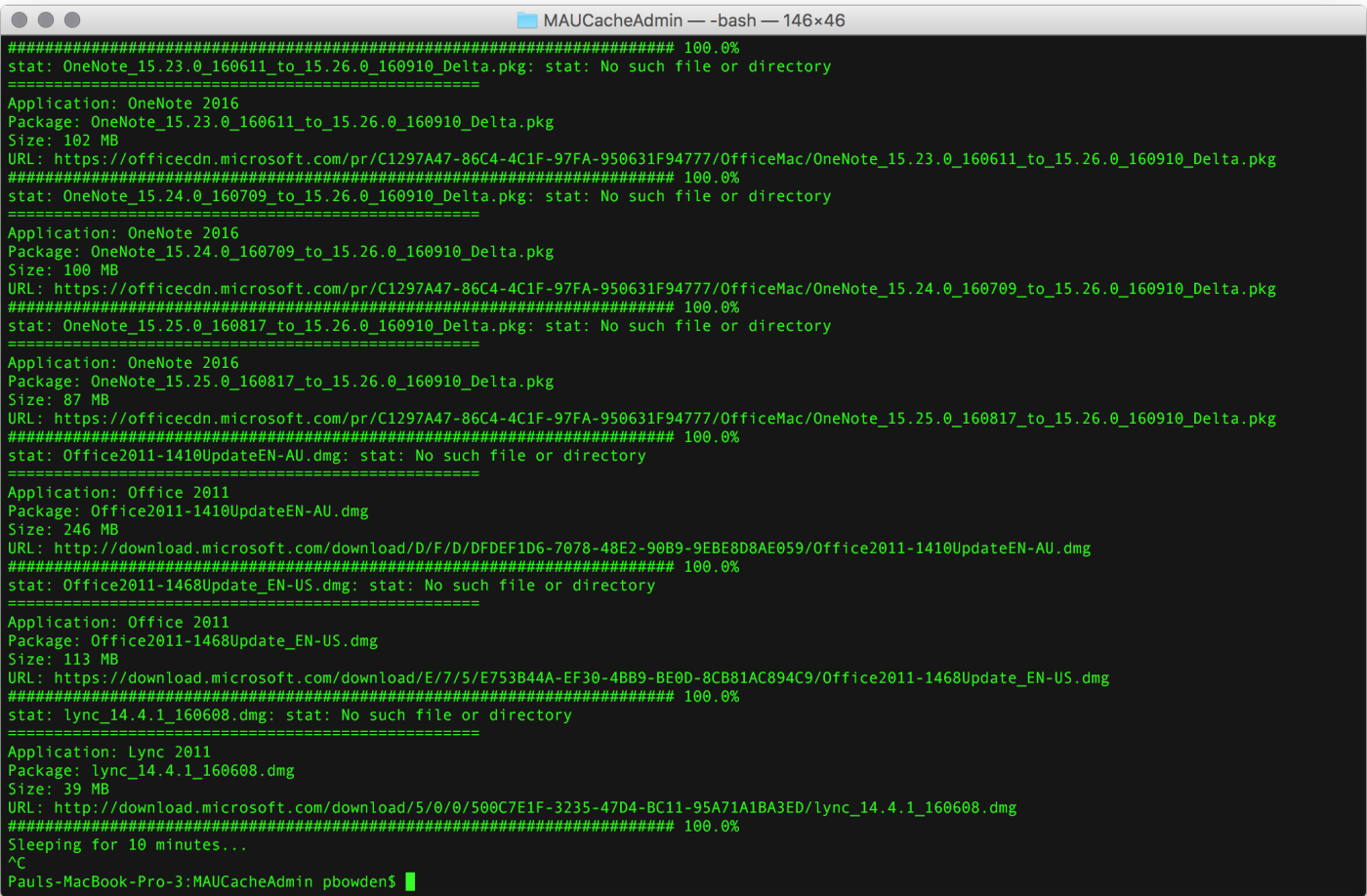
There are three components that make up the UpdateCache solution:

1. The MAUCacheAdmin tool which copies files from Microsoft’s CDN to a folder of your choice.
2. An HTTP/HTTPS web service that exposes your folder to clients on your network.
3. A configuration change to the MAU client on each users’ machine.

The MAUCacheAdmin tool

The ‘MAUCacheAdmin’ tool can be used to copy both collateral and update packages from the Microsoft CDN to a folder of your choice. The latest version of MAUCacheAdmin, which is a bash script, can be downloaded from <https://github.com/pbowden-msft/MAUCacheAdmin>

By default, the MAUCacheAdmin tool checks the CDN and downloads updates just once. If you wish to run the tool in a loop use the --CheckInterval command-line parameter. All available update packages are downloaded into the root of the folder that you specify with the --CachePath command-line parameter. Only production quality builds are downloaded by the tool. For Office 2016 applications, the tool will download both ‘full’ updates, and ‘delta’ updates for the previous three releases. A typical monthly update for all apps will consume ~8GB of disk space.



HTTP/HTTPS web service

You will need to expose the CachePath folder as part of an HTTP or HTTPS server. MAU has no dependency on the operating system or version of the web host. Any HTTP server, including Apache, Internet Information Service (IIS), and even python’s SimpleWebServer is capable of hosting MAU content. The only requirement that MAU has is that the server must return a 404 response if it doesn’t have a copy of the requested package.

NOTE: The web service must use the standard ports of 80 and 443. MAU does not support custom port definitions.

Configuring the MAU client to use a local server

Once your web server is deployed, you must configure each users’ MAU client to prefer the local service over the CDN. You can use Configuration Profiles to deploy these overrides, or simply use the defaults command-line tool to set local preferences:

```
defaults write com.microsoft.autoupdate2 UpdateCache -string 'https://server/folder/'
```

IMPORTANT: Ensure that a trailing slash is used when specifying the value for the UpdateCache preference. This is mandatory.

In this scenario, MAU will still use the collateral on the Microsoft CDN to detect updates, but before downloading those update packages over the Internet from the CDN, it will first check the UpdateCache server. If the UpdateCache server has a local copy of the update, a 200 response will be sent to the client and MAU will obtain its update from the local server. If the server returns a 404 (not found) response, MAU will fall-back to downloading the package from the CDN.

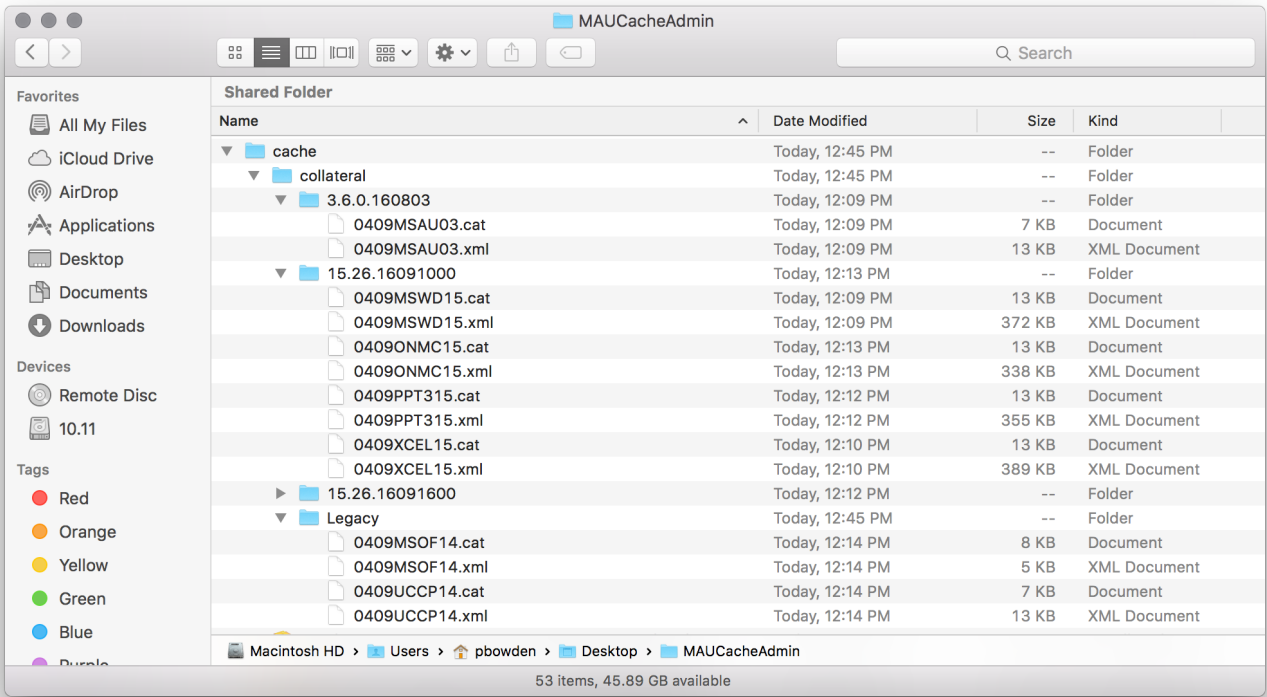
Scenario 2: Deploying a Custom ‘ManifestServer’ Service

There are three components that make up the ManifestServer solution:

- 1. Obtaining copies of MAU’s collateral
- 2. An HTTP/HTTPS web service that exposes the collateral to clients on your network.
- 3. A configuration change to the MAU client on each users’ machine.

Obtaining MAU collateral

You can use the MAUCacheAdmin tool to obtain collateral from the CDN. On each checking cycle, MAUCacheAdmin will download the latest collateral and place it in the ‘collateral’ sub-folder of the CachePath. For Office 2016 for Mac apps, the XML and CAT files are stored under a per-version folder. Office 2011 collateral is stored under a sub-folder called ‘Legacy’.



You can also use a tool such as curl to download copies of collateral from Microsoft’s servers to your custom server. For example:

```
curl -# --output --url "https://officecdn.microsoft.com/pr/4B2D7701-0A4F-49C8-B4CB-0C2D4043F51F/OfficeMac/0409PPT315.{xml,cat}"
```

Finally, you can find archives of Office 2016 application collateral at <http://macadmins.software>. Simply download the DMG relative to the ‘maximum’ version you want MAU to see.

HTTP/HTTPS web service

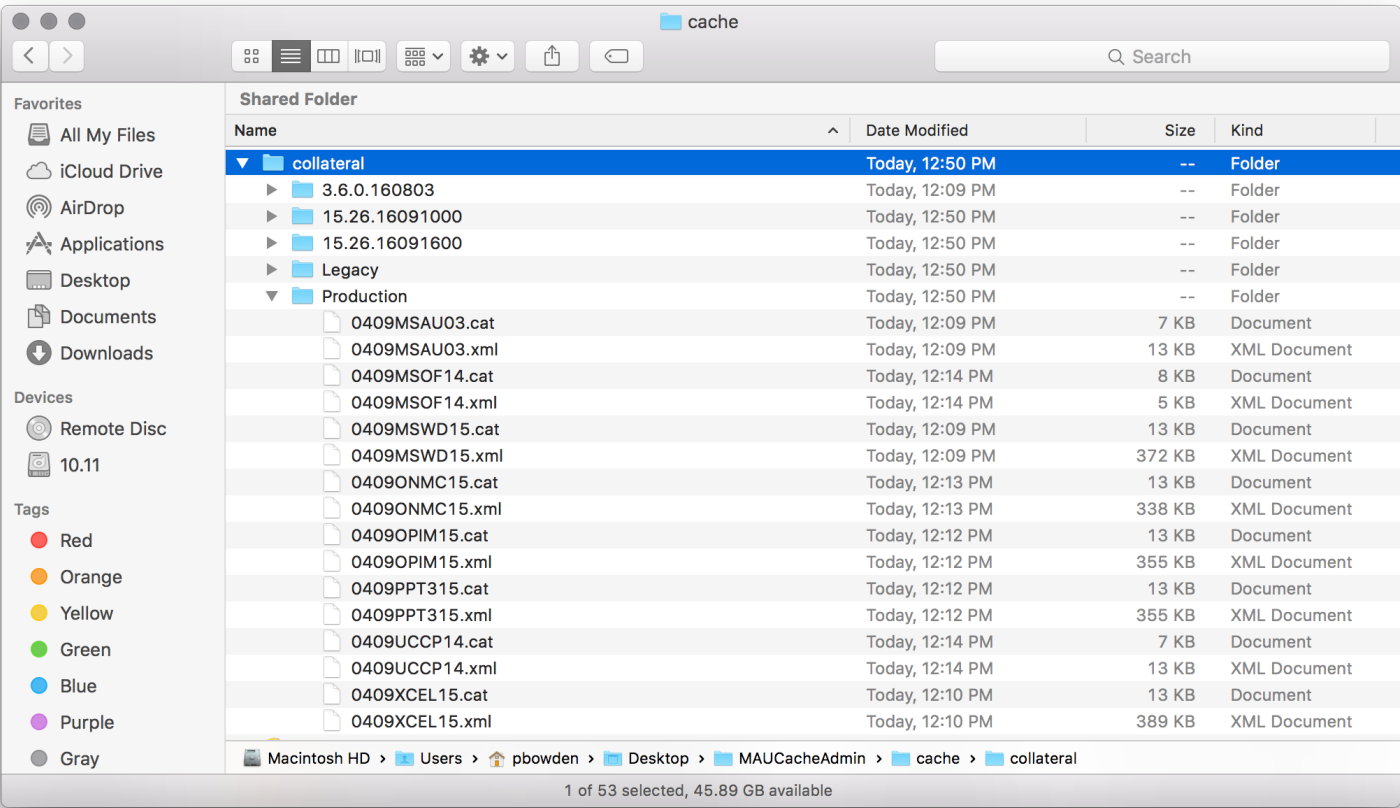
You will need to expose your collateral folder as part of an HTTP or HTTPS server. MAU has no dependency on the operating system or version of the web host. Any HTTP server, including Apache, Internet Information Service (IIS), and even python’s SimpleWebServer is capable of hosting MAU content. It is recommended that you create a folder called ‘Production’ and drag the relevant application collateral into that folder. This flat folder of application collateral is what the MAU client will use to determine in an update is available. Think of the versioned folders as a long-term archive.

You must deploy collateral for all applications to your custom manifest server, not just the apps that you wish to control. For example, if you don’t deploy 0409UCCP14 collateral, MAU will not be able to check for Lync updates.

If your custom manifest server is on your corporate network, users who take their machines home may not have direct access to the web server and MAU will not be able to check for updates.

If you are deploying both an UpdateCache and custom ManifestServer, you can use either the same web server, or different servers – it’s your choice.

NOTE: The web service must use the standard ports of 80 and 443. MAU does not support custom port definitions.



Configuring the MAU client to use custom manifests

In addition to Microsoft-defined update channels, MAU supports a custom channel where you can specify your own manifest server. You can use Configuration Profiles to deploy these overrides, or simply use the defaults command-line tool to set local preferences:

```
defaults write com.microsoft.autoupdate2 ChannelName -string 'Custom'
defaults write com.microsoft.autoupdate2 ManifestServer -string 'https://server/folder/'
```

IMPORTANT: Ensure that a trailing slash is used when specifying the value for the ManifestServer preference. This is mandatory.

If MAU has been configured to use a custom manifest server, it will use that exact path as the single authority of updates. If your custom manifest server is down or non-functional, MAU will report that the update server could not be reached. It will not fail-through to Microsoft’s servers.

You cannot use MAU to deploy custom applications. Attempting to alter the manifest (XML) file will cause a file signature change, which MAU will reject.

Document History

Date/Version	Changes
October 6, 2016 – 1.0	Initial version, based on contents from ‘Implementing a Custom Manifest Server for MAU’
October 21, 2016 – 1.1	Revised the MAUCacheAdmin section as the tool now just runs once by default