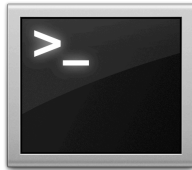


InstaDMG 1.6rc1



Super-Ultra-Quickstart:

Open Terminal,



and paste:

```
svn checkout http://instadmog.googlecode.com/svn/trunk instadmog
```

Insert a *retail-box (important)* OS 10.6 (or 10.6.3) installer disk

Paste the following, entering your (admin) password when prompted:

```
sudo ./instadmog/AddOns/InstaUp2Date/importDisk.py --automatic
```

Let it process for about 45 minutes, and finally:

```
sudo ./instadmog/AddOns/InstaUp2Date/instaUp2Date.py 10.6_vanilla --process
```

Collect a fully patched, “10.6.5 Vanilla.dmg”, in `./instadmog/OutputFiles`

Of course, there's much more to it than that!

Introduction - Where It Fits, What It Is

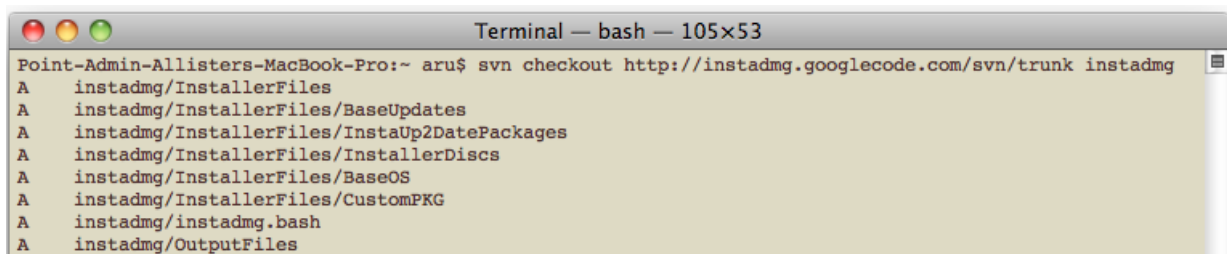
When setting up many new computers at once or refreshing existing workstations, it is of the utmost importance to have a known baseline you can count on. Out of the desire to ‘slipstream’ (loosely speaking) updates, software, and/or other enhancements into a baseline image that is hardware independent, InstaDMG was born. It creates an image that can be customized through the addition of installer packages (which enables a ‘modular’ approach) with the defining feature that the resulting image has never been booted.

Once we understand this specific use, it should be appreciated as part of the “Preparation” step according to Apple’s [Best Practices for Client Management\(pdf\)](#). Being only the part that creates an image, it is easy to downplay its role if we don’t appreciate the many deployment tools that can push out the resulting image in an optimized manner. asr is a binary that powers many of those tools, which is used to create and deploy images at the block-level (instead of file-level), which results in very fast restores.

Having become accustomed to the basic, straightforward way of manually configuring a workstation, we will need to take a different approach to get the changes we want to be applied to our image. Many of the common questions of how to effectively apply those customizations, that would otherwise have to be handled during the Maintenance and Control/Monitoring steps, have been solved by the community on the [afp548.com forums](#). We won’t be covering all of the techniques available, instead focusing on an overview of all of the parts of InstaDMG that can help sysadmins get the baseline they need.

Before we move forward, let’s walk through the quickstart InstaDMG usage scenario above.

Figure A.



```
Terminal — bash — 105x53
Point-Admin-Allisters-MacBook-Pro:~ aru$ svn checkout http://instadmg.googlecode.com/svn/trunk instadmg
A   instadmg/InstallerFiles
A   instadmg/InstallerFiles/BaseUpdates
A   instadmg/InstallerFiles/InstaUp2DatePackages
A   instadmg/InstallerFiles/InstallerDiscs
A   instadmg/InstallerFiles/BaseOS
A   instadmg/InstallerFiles/CustomPKG
A   instadmg/instadmg.bash
A   instadmg/OutputFiles
```

Ride the Bullet Build Train

“svn checkout http://instadmg.googlecode.com/svn/trunk instadmg”

This pulls down the most recent version of the project into a newly-created folder called “instadmg” - it finishes in a matter of seconds (similar to Figure A, above), producing many lines of output and creating a very specific directory structure (Figure B, below) in your home folder.

```
“sudo ./instadm/AddOns/InstaUp2Date/importDisk.py --automatic”
```

After prompting you for your (currently-logged-in admin) password, this prepares the inserted installer disk (DVD) by creating a dmg of it and placing that in the `./instadm/InstallerFiles/InstallerDiscs` folder. The “automatic” option makes it skip prompting you for which disk you’d like to import, since we’re assuming there is only one installer disk mounted. Expect this step to take around 45 minutes or so.

```
“sudo ./instadm/AddOns/InstaUp2Date/instaUp2Date.py 10.6_vanilla --process”
```

The important thing to notice is the `--process` option, which triggers the transition from the `instaUp2Date` functionality (which pulls down the updates specified in the `10.6_vanilla.catalog` file,) to the running of our key player, the `instadm` bash script. The end result (after over an hour, surely enough time to grab a beverage,) will be a ready-to-restore dmg in the `./instadm/OutputFiles` folder.

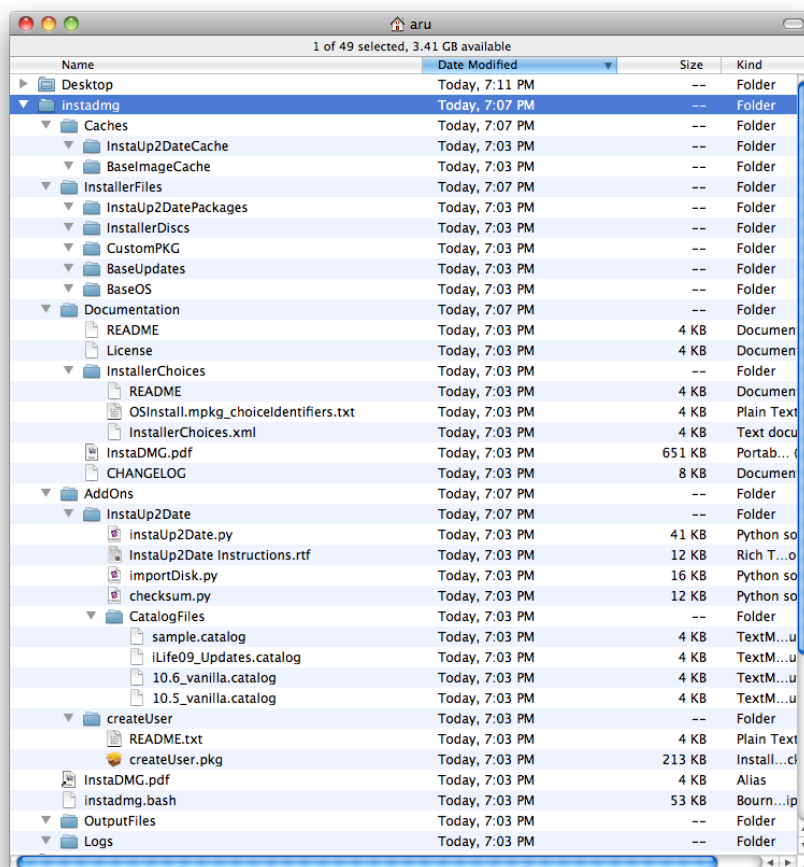


Figure B.

Wait, what's InstaUp2Date?

It's a cause of confusion for some that `InstaUp2Date` is a project developed in parallel with `InstaDMG`, so we'll cover its benefits and use. Among other reasons, the wish to collaborate on a list of Apple patches to include in any generic image inspired the creation of `InstaUp2Date`. Sysadmins appreciate the utility of feeding a configuration file into the process

(rather than merely passing a lot of flags and then parsing a log on the other end), and it acts as a documentation source to account for what went into an image.

In our quickstart earlier, the InstaUp2Date python program takes its instructions from the 10.6_vanilla catalog file, and then proceeds to download the updates listed and place them in a cache folder. Checking that you received the updates you wanted “as advertised” (i.e. not corrupted or altered) is done by verifying the sha1 checksum, included in the catalog file. Symbolic links (a.k.a. aliases or shortcuts, loosely speaking) are created to point from where InstaDMG would expect packages in the directory structure to where the files that have been cached. InstaDMG is then started up, via being called by the --process flag, at which point instaup2date.py’s most visible function is complete.

Check out `./instadm/AddOns/InstaUp2Date/Catalogs/sample.catalog` and more comprehensive instructions can be found in the [readme](#)

Procedurals

If we’re not utilizing InstaUp2Date to take care of the moving around of packages, we need to sort them ourselves. Adding packages (or mpkgs) to our image after the OS is installed, and making sure iLife gets installed before its updates are applied, means we need to get a handle on what order things are installed in. Even though many of these packages will be updates we get from Apple, it’s a good idea to separate out the ones that deal with the core functionality of the OS (in `./instadm/InstallerFiles/BaseUpdates`) and others, including software or peripheral-specific ones (which belong in `./instadm/InstallerFiles/CustomPKG`).

Manually running updates for the OS (or other products like iLife or Microsoft Office) with their native update software can make it easy to see exactly what order updates should be applied in. In the GUI you can check by going into System Preferences -> Software Update and looking under the Installed Updates tab. Once we’ve assembled the update packages, we place them in numbered folders (01, 02, etc.) with one package per folder to be specific about their ordering.

On to the Good Stuff

This is by no means an exhaustive look into InstaDMG’s real-world usage, but the rest of this guide will be an overview of “how do I do that?”, which will hopefully include some “I didn’t know it could do THIS!” We won’t step through every flag and option (there will be no mention of flags until the “rev the engine” section,) for the sake of touching on all the shiny bits and avoiding the sharp, pointy ones.

Brass tacks:

The first fundamental part of all of this is the Operating System itself, referred to as the BaseOS. You are absolutely going to want to specify your own ‘answer’ file, which essentially checks boxes for you as if you were using the GUI installer process and making selections in the “customize” interface yourself. Examples are included for Leopard in the `./instadm/Documentation/InstallerChoices` folder. Put the `InstallerChoices.xml` file in the folder with the OS (`./instadm/InstallerFiles/BaseOS`) and `InstaDMG` will evaluate what to install based upon its contents.

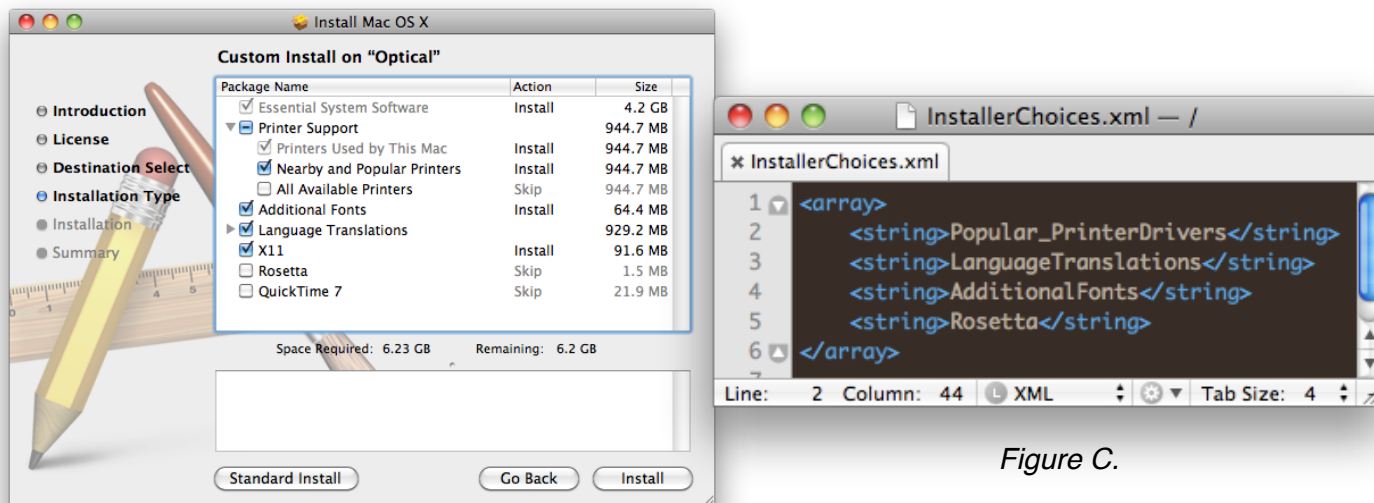


Figure C.

Unless, of course, you **do** need those extras

Why and How this here answer file business works:

When you’re de-selecting boxes in the GUI interface, it tells the installer to modify what to include in comparison to its Standard Install (which would be everything presented by default according to how Apple built the mpkg). Taking a step back, this customization is only possible when working with a mpkg. Since we can bundle many pkg installs into one mpkg, when the mpkg is built you may be able to specify certain things to leave out or include, whether or not they are even visible in the GUI installer.

The `InstallerChoices` readme has a [link](#) to a more thorough process, but a one-liner to get started with is:

```
sudo installer -showChoicesXML -pkg $PATH_TO_MPKG
```

Where `PATH_TO_MPKG` would be

”/Volumes/Mac\ OS\ X\ Install\ DVD/System/Installation/Packages/OSInstall.mpkg” in the case of the retail OS install itself. Following the instructions in that forum post above for Snow Leopard should end you up with something like Figure C., above.

The particularly useful thing about learning how to take advantage of an answer file is that you can use the same process for other mpkg's, like Office2008 and iLife 09, and then add the xml to the folder with the installer (*this is all site licensed software we're working with, of course.*) Use the installer GUI as your guide, and give yourself a little background by looking at the full choicesXML for context, just in case the naming of the choices aren't clear.

Benefits Out-Of-The-Box, and Added On

In an effort to make builds happen faster on subsequent runs, the base image (after installerChoices are evaluated) gets cached in `./instadm/Caches/BaseOS`. InstaDMG will then look there from that point on, and subsequent runs skip that initial install step "for free".

If you haven't already, you should look into DeployStudio, PSU Blast Image Config, or even just `asr` at the command line to restore your image. *The usual warnings apply, this will wipe a disk clean before laying down your image - although all these tools are free, caveat emptor.* You'll notice the restore process can be as fast as 3 minutes for a lean (< 6GB) image. Here's a one-liner for `asr` to get you going:

```
sudo asr restore --source ./instadm/OutputFiles/10-11-08.dmg --target /Volumes/  
Destination --erase --verbose --noprompt --noverify --buffers 1 --buffersize 32m  
--puppetstrings
```

Added-on shiny bits:

Included in the AddOns directory is a `createUser` package, which enables you to place a fresh, never logged in user account on the image. Almost everything you'd specify in System Preferences -> Accounts when creating a new user can be customized here, and a tool to use which can generate an obscured password. This goes hand-in-hand with a very spare package that can be obtained from afp548.com's MyDownloads section, called `clearReg`. This tricks the setup assistant to believe it has already run, along with the prompt for registration.

Revisiting `InstaUp2Date`, we'll touch on another helper file, `checksum.py`. This allows us to refer to custom pkgs we'd like to add in catalog files in the correct format (its output should be formatted with tabs exactly as required). Every time the 'fingerprint' or makeup of your package changes you will need to re-run this against your pkg and then update your catalog file accordingly. This is achieved with the following:

```
/instadm/AddOns/InstaUp2Date/checksum.py $PATH_TO_PKG
```

Where `$PATH_TO_PKG` usually lives in `/instadm/InstallerFiles/InstaUp2DatePackages`

Problems with certain installers? You can always just punt

If getting your software to play well with InstaDMG (or deployment in general) isn't working for you, there are packaging tools that can 'capture' (like a snapshot) the state of a machine before and after install so you can re-package the differences. There are varying levels of success and complexity with the tools out there so we won't cover this topic in more depth, but a general recommendation is to run the software once before considering the capture stage complete.

How to rev the engine

Once we've got the workflow down, it's all about optimization. Disk image creation is a heavy I/O process, so CPU and RAM don't play as much of a factor (although only having 1GB RAM is painful in general). There is one step in the process that can be skipped, and other helpful flags I'll demonstrate below:

```
sudo ./instadmg/instadmg.bash -f -t /Volumes/stripedRAID0 -o /Volumes/stripedRAID0
-m Dev_10-6-5 -n Restored
```

F denotes "non-paranoid mode", which will therefore skip the verification of image checksums. The subsequent flags have the effect of multiplying the 'spindles' doing the work; our InstaDMG directory structure can be housed on a separate disk for the sole purpose of reading the packages (these days that would be a SSD with its superior sequential reads nearly saturating the SATA bus) and another location designated by the T flag (in this example a RAID0 volume), is utilized for both writing the intermediate image into its root directory (instead of the default /tmp of your currently booted volume), and putting the final asr-ready output (-o, instead of ./instadmg/OutputFiles/).

The M and N flags are convenient ways to help you tell your images apart by setting what you want the resulting asr image to be named, both before restoration (-m, which would result in "Dev_10-6-5.dmg" instead of the default naming "10-11-17.dmg" for November 17th, 2010) and after (-n, so the resulting partition name would be "Restored" instead of the default "InstaDMG".)

These are scripts you can open and look at, so feel free to read through all the options, or pass the --help flag to it. And please post on the forum at afp548.com with questions! Questions/feedback about this guide? instadmg.docs@gmail.com

Roll Credits, Thanks

To you, for reading this far,



contributors past and future,

AFP548
.COM

**Mr. Kuehn, Mr. Wisenbaker, Mr. Fergus, Mr. Akins, Mr. Meyer,
 (and Mr. Larizza, and Mr. Walck, and Mr. van Bochoven, and...)**