InstaDMG 1.6b2

Super-Ultra-Quickstart:

Open Terminal,  **and paste:**

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

Insert a retail-box copy of the OS 10.6 install disk

Paste the following, entering your password when prompted:

```
sudo ./instadmg/AddOns/InstaUp2Date/importDisk.py --automatic --legacy
```

Let it process for about 45 minutes, and finally:

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

Collect a fully patched,"10.6.2 Vanilla.dmg", in ./instadmg/OutputFiles!

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

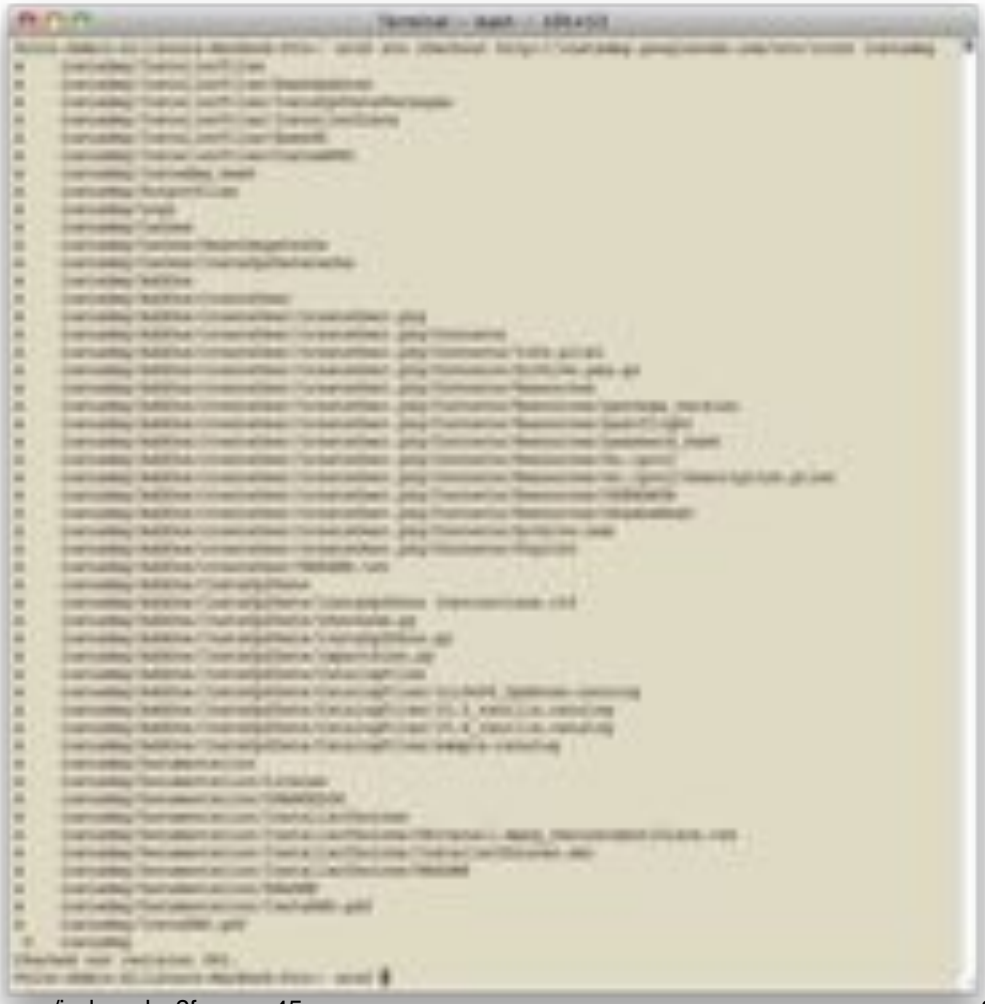
Introduction - Where It Fits, What It Is

When setting up many computers at a time or refreshing existing workstations, having a repeatable, consistent method with which to set the initial configuration ensures a baseline you can count on. Many Mac management tools operate once there's already a working image, like Radmind, Puppet and Apple Remote Desktop, each with different uses(moving parts) and goals.

asr is a tool to create and deploy images in a optimized manner(block-level instead of file-level,) which can be used in some cases with Disk Utility's restore function. Another straightforward way to get started with deployment when you have many workstations that are the same model is by building a 'golden master' on one of them, and then restoring to the rest. This has it's limits(drivers specific to one model, etc.) and drawbacks(e.g. human intervention.)

InstaDMG uses a bash script to create customized images for deployment via asr with the defining feature that they have never been booted. Additionally, and as the result of hard work and design decisions, these images can work well with a wide variety of hardware, software and settings. The biggest moving parts are the payloads and scripts that can be included in pkgs and mpkgs, and customizations passed to the installer process via xml files. Before we move forward, let's walk through the quickstart InstaDMG usage scenario above.

Figure A.



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(as seen in Figure A above,) producing 51 lines of output and a very specific directory structure(Figure B below) in your home folder.

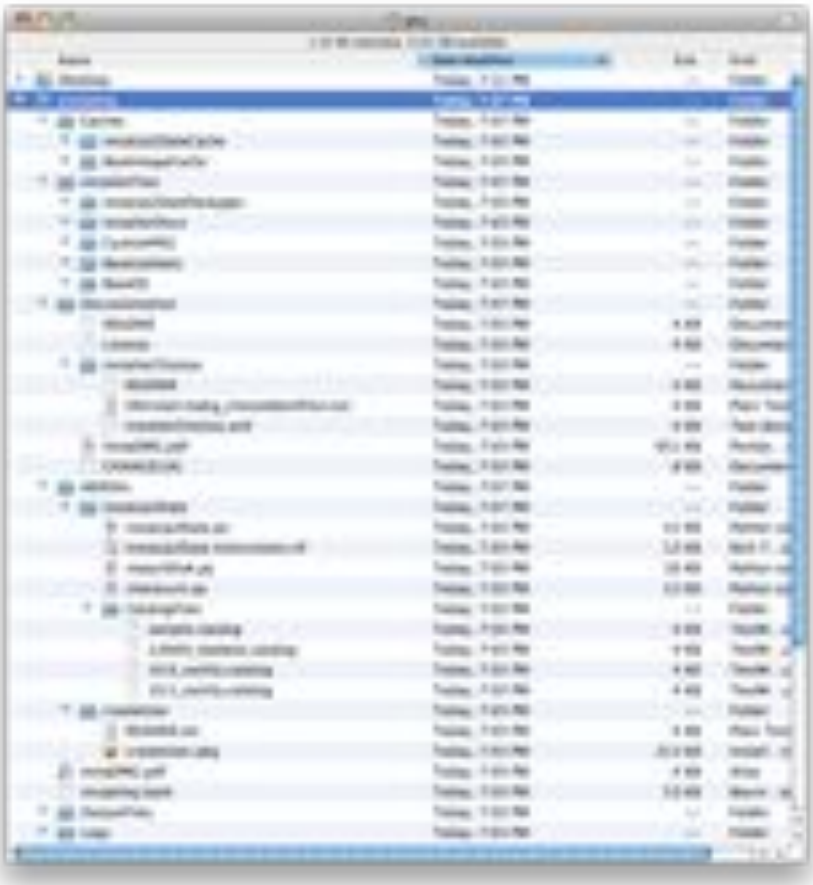
"`sudo ./instadmg/AddOns/InstaUp2Date/importDisk.py --automatic --legacy`"

After prompting you for your (currently-logged-in admin's) password, this prepares the inserted installer disk(DVD) by creating a dmg of it and placing that in the ./instadmg/InstallerFiles/BaseOS folder. The options at the end make it skip prompting you for what OS(client or server) the disk is, and uses "legacy" naming and location for the created image. Expect this step to take around 45 minutes or so.

"`sudo ./instadmg/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 instadmg 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 ./instadmg/OutputFiles folder.

Figure B.



Side Tangent: InstaUp2Date

Among other reasons, the wish to automate the pulling down of updates from Apple's servers inspired the creation of InstaUp2Date. It takes instructions from catalog files containing a certain format and order with which to download updates and place them in a cache directory. You can use these catalogs for reference purposes, to collaborate with others (as evidenced by the maintainers of the 'vanilla' catalog files,) and to add other items from various locations(with caveats.) Another nicety of using InstaUp2Date is you can clearly specify the Output Volume Name(what the volume your imaging will be named, e.g. the ubiquitous "Macintosh HD"), and the Output File Name, which is what the resulting dmg is named.

Checking that you received the updates you wanted "as advertised"(i.e. not corrupted) is done by verifying the sha1 checksum. For custom packages, this can be generated via the checksum.py add-on. This is a separate project, so for now we'll move on.

Check out the sample.catalog and readme for more info

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 ./instadmg/InstallerFiles/BaseUpdates) and others, including model or peripheral-specific ones(which belong in ./instadmg/InstallerFiles/CustomPKG).

Running through updates for the OS or other product with their native mechanisms 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, there is the school of thought that making numbered folders(01 through 99 if you have that many, or just one digit if less than 10) is only necessary to specify what has to happen later in the install process, although one package per folder is the only way to be explicit and is considered best practice.

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 with 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 OS 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 ./instadmg/Documentation/InstallerChoices folder. Put the InstallerChoices.xml file in the folder with the OS (./instadmg/InstallerFiles/BaseOS) and InstaDMG will evaluate what to install based upon its contents.

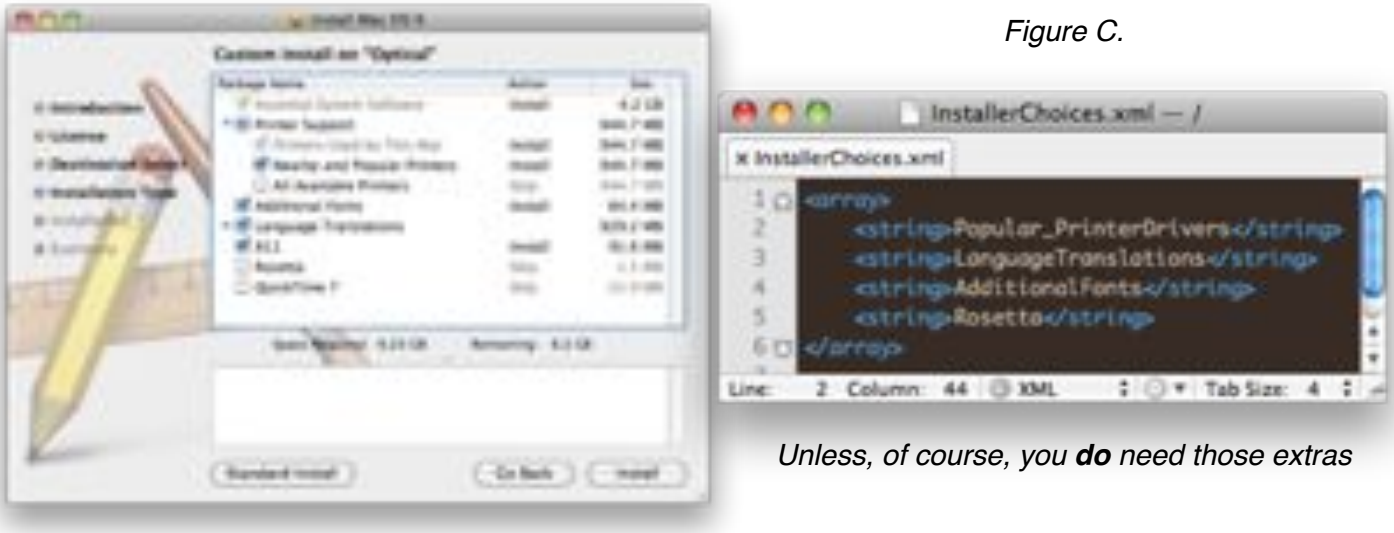


Figure C.

*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 whoever 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, wether or not they are even visible in the GUI.

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 OS itself. Following the instructions in that forum post linked by the readme 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.) Just use the 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 addition to InstaUp2Date which we briefly mentioned above, there are other 'AddOns' that we'll touch upon in just a bit, but first let's discuss the big picture. In an effort to make your build happen faster on subsequent runs, the base image(after installerChoices are evaluated) gets cached in a location that it will look for from then on, and you get much faster subsequent run times "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 image. Here's a one-liner for asr to get you going:

```
sudo asr restore --source ./instadmg/OutputFiles/10-3-7.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 you can obscure the password for safety. This goes hand-in-hand with another package you can either make(with PackageMaker, included with Xcode and ARD) or get 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.

What both of these packages have in common is their usage of the postflight script, which opens up a wide range of possibilities. While we won't go further into this deep topic, let's just touch on one enhancement as an example:

```
#use the 'kickstart' function of the apple remote desktop agent to turn access on
for the just-created admin

/System/Library/CoreServices/RemoteManagement/ARDAgent.app/Contents/Resources/kicks
tart -activate -configure -access -on -users $shortname -privs -all -restart -agent
-target "$3"
```

Adding the lines above(comments for the win) to the bottom of the postflight script included in createUser(*"Show Package Contents" when right-clicking the pkg, then open ./Contents/Resources/postflight in a text editor*) allows ARD access to be enabled for that(presumed admin) user the moment the booted image gets to login screen. Batteries included.

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 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 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 optimizing. Disk image creation is a heavy I/O process, so CPU and RAM don't play as much of a factor(though only having 1GB 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_10-6-2 -n Restored
```

F denotes "non-paranoid mode", which will therefore skip the verification of image checksums. The T and O flags have the effect of multiplying the 'spindles' doing the work; a separate disk where instaDMG lives for reading the packages(these days that would be a SSD with its superior sequential reads nearly saturating the SATA bus) and another location, in this example an external RAID0 volume we could attach over eSATA or fibre, for both writing the intermediate image into its /tmp directory(`-t`, instead of the default /tmp of your currently booted volume), and putting the final asr-ready output(`-o`, instead of ./instadmg/OutputFiles/).

As is an option in InstaUp2Date's catalog files, 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_10-6-2.dmg" instead of the default naming"10-3-9.dmg" for March 9th, 2010) and after(`-n`, so the resulting partition name would be "Restored" instead of the default "InstaDMG".)

With the hardware options & flags above, one can create an image with iPhoto and Office 08' using a cached BaseOS in less than 25 minutes. This is a bash script you can open and look at, so feel free to read through all the options. 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,

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