dmgbuild Documentation

Release 1.0.0

Alastair Houghton

February 17, 2014

This document refers to version 1.0.0

CHAPTER 1

What is this?

`dmgbuild` is a command line tool to create Mac OS X disk images (aka `.dmg` files). While it is possible to create disk images easily enough from the command line using the `hdiutil` program that ships with Mac OS X, there is no easy way to configure the appearance of the resulting disk image when the user opens it. Some people have used AppleScript to automate Finder to adjust the appearance, but since Finder saves its `.DS_Store` files asynchronously, it is hard to guarantee that the changes will actually be saved when you want them to be. It also means that you need a GUI session, with Finder running, in order to build your disk image.

`dmgbuild` does not rely on Finder; nor does it rely on deprecated APIs (like the Alias Manager functions). Instead, it uses the `ds_store` and `mac_alias` Python modules, which know how to construct the relevant data in Python code.

Contents:

1.1 Usage

Typical usage looks like this:

```
dmgbuild -c settings.py "Volume Name" output.dmg
```

The (optional) `settings.py` file specifies the attributes that control the appearance of the disk image. The other arguments give the volume name and the name of the output file respectively.

`dmgbuild` also accepts arguments of the form `-D key=value`; these can be used from within the settings file, for instance so that you can write a single settings file for multiple disk images, or so that you can easily alter settings at build time.

1.2 Settings

`dmgbuild` accepts, as one of its arguments, a settings file. This is in fact a Python script, which means anything you can do in Python code, you can do in your settings file. This makes it easy for you to customise the behaviour of `dmgbuild`.

Each of the available settings is documented below; all of them are optional; the `dmgbuild` program has defaults for those that matter. The default values *are* visible from within your settings file, if you want to examine or alter them rather than replacing them completely.

If any `-D key=value` settings have been made on the command line, they are visible in a dictionary named `defines` within the settings file.

1.2.1 Disk Image Settings

filename
If defined, overrides the output filename specified on the command line. The command line value is the default value.

volume_name
If defined, overrides the volume name specified on the command line, which is the default value.

format
Specifies the format code for the final output disk image. Must be one of the types supported by `hdiutil` on the build system; currently the list includes

Code	Meaning
UDRO	Read-only
UDCO	Compressed (ADC)
UDZO	Compressed (gzip)
UDBZ	Compressed (bzip2)
UFBI	Entire device
IPOD	iPod image
UDxx	UDIF stub
UDSB	Sparse bundle
UDSP	Sparse
UDRW	Read/write
UDTO	DVD/CD master
DC42	Disk Copy 4.2
RdWr	NDIF read/write
Rdxx	NDIF read-only
ROCo	NDIF Compressed
Rken	NDIF Compressed (KenCode)

For disk images you intend to distribute over the Internet, you should probably stick to 'UDZO' and 'UDBZ'.

size
Specifies the size of the filesystem within the image. You should set this large enough to hold the files you intend to copy into the image. The syntax is the same as for the `-size` argument to `hdiutil`, i.e. you can use the suffixes 'b', 'k', 'm', 'g', 't', 'p' and 'e' for bytes, kilobytes, megabytes, gigabytes, terabytes, exabytes and petabytes respectively.

Defaults to '100M'

1.2.2 Content Settings

files
A list of files (or folders) to copy into the image. Each of these is copied to the root of the image; folders are copied recursively. e.g.:

```
files = [ '/Applications/TextEdit.app' ]
```

symlinks
A dictionary specifying symbolic links to create in the image. For example:

```
symlinks = { 'Applications': '/Applications' }
```

icon
Specifies the path of an icon file to copy to the volume. You can either specify this, or as an alternative you can use the `badge_icon` setting.

badge_icon
As an alternative to the above, if you set *badge_icon* to the path of an icon file or image, it will be used to badge the system's standard external disk icon. This is a convenient way to construct a suitable icon from your application's icon, e.g.:

```
badge_icon = '/Applications/TextEdit.app/Contents/Resources/Edit.icns'
```

icon_locations
A dictionary specifying the co-ordinates of items in the root directory of the disk image, where the keys are filenames and the values are (x, y) tuples. e.g.:

```
icon_locations = {
    'TextEdit.app': (100, 100),
    'Applications': (300, 100)
}
```

1.2.3 Window Settings

background
A string containing any of the following:

Example	Meaning
#3344ff	Web-style RGB color
#34f	Web-style RGB color, short form (#34f = #3344ff)
rgb(1,0,0)	RGB color, each value is between 0 and 1
hsl(120,1,.5)	HSL (Hue Saturation Lightness) color
hwb(300,0,0)	HWB (Hue Whiteness Blackness) color
cmyk(0,1,0,0)	CMYK (Cyan Magenta Yellow Black) color
goldenrod	X11/SVG named color
builtin-arrow	A simple blue arrow image (retina enabled)
/foo/bar/baz.png	The path to an image file

The hue component in `hsl()` and `hwb()` may include a unit; it defaults to degrees ('deg'), but also supports radians ('rad') and gradians ('grad' or 'gon').

Other color components may be expressed either in the range 0 to 1, or as percentages (e.g. 60% is equivalent to 0.6).

For no background, specify `None` instead of a string value.

show_status_bar
show_tab_view
show_toolbar
show_pathbar
show_sidebar
Each of the above controls the display of one of the standard window elements. All of them default to `False`.

sidebar_width
The width of the Finder sidebar.

window_rect
The position of the window in `((x, y), (w, h))` format, with y co-ordinates running from bottom to top. The Finder makes sure that the window will be on the user's display, so if you want your window at the top left of the display you could use `(0, 100000)` as the x, y co-ordinates. Unfortunately it doesn't appear to be possible to position the window relative to the top left or relative to the centre of the user's screen.

default_view
The default view for the window; should be a string containing one of:

View name
icon-view
list-view
column-view
coverflow

show_icon_preview
Whether or not to show icon previews for the contents of the disk image (defaults to `False`)

include_icon_view_settings
include_list_view_settings
Set these to `True` to force inclusion of the icon/list view settings respectively. By default, `dmgbuild` will only include settings for the default view type.

1.2.4 Icon View Settings

arrange_by
If set, indicates that the Finder should arrange the icons in the icon view according to the specified field. Allowable settings are:

Field name
name
date-modified
date-created
date-added
date-last-opened
size
kind
label

Any other value disables automatic icon arrangement (which is the default, since the main use-case for `dmgbuild` is building application distribution images, where icon positioning is an important part of the design).

grid_offset
Specifies the grid offset for automatic arrangement.

grid-spacing
Specifies the grid spacing for automatic arrangement.

scroll_position
An (x, y) tuple specifying the scroll position; this is only relevant if you position icons outside of the window area.

label_pos
Specifies the position of the icons' labels. Choose 'bottom' or 'right' (defaults to 'bottom').

text_size
Specifies the point size of the label text. Default is 16pt.

icon_size
Specifies the size of icon to use. Default is 128pt.

icon_locations
If `arrange_by` is not set, a dictionary mapping the names of items in the root of the volume to an (x, y) tuple specifying their location in points.

1.2.5 List View Settings

In list view, the following columns are available:

Field name
name
date-modified
date-created
date-added
date-last-opened
size
kind
label
version
comments

list_icon_size
Sets the size of the icon in list view. Default is 16pt.

list_text_size
Sets the size of the text in list view. Default is 12pt.

list_scroll_position
Specifies the scroll position, assuming there are enough items to make the view scroll.

list_sort_by
Specifies which column the Finder should sort the display by. Defaults to 'name'.

list_use_relative_dates
If `True`, formats dates using words like "Today" or "Yesterday" where possible; otherwise they will be displayed as a full date. Defaults to `True`.

list_calculate_all_sizes
If `True`, forces the Finder to compute all of the item sizes; normally this is set to `False` because it can be expensive calculating the sizes of deeply nested folders. Defaults to `False`.

list_columns
A list or tuple of strings containing the names of columns, in the order you want them to appear.

list_column_widths
A dictionary specifying the width, in points, for each of the columns. There are default widths for every column, so you may not need to set this variable in practice.

list_column_sort_directions
A dictionary specifying the sort direction (either 'ascending', or 'descending') for each column. Again, there are individual defaults for each column, so you may not need to touch this unless you wish to override the default behaviour.

1.3 Example Settings File

Below is a copy of the example settings file that you can find in the source distribution in the folder "example".

```
# -*- coding: utf-8 -*-
from __future__ import unicode_literals

import biplist
import os.path
```

```
#
# Example settings file for dmgbuild
#

# Use like this: dmgbuild -s settings.py "Test Volume" test.dmg

# You can actually use this file for your own application (not just TextEdit)
# by doing e.g.
#
#   dmgbuild -s settings.py -D app=/path/to/My.app "My Application" MyApp.dmg

# .. Useful stuff ..............................................................

application = defines.get('app', '/Applications/TextEdit.app')
appname = os.path.basename(application)

def icon_from_app(app_path):
    plist_path = os.path.join(app_path, 'Contents', 'Info.plist')
    plist = biplist.readPlist(plist_path)
    icon_name = plist['CFBundleIconFile']
    icon_root,icon_ext = os.path.splitext(icon_name)
    if not icon_ext:
        icon_ext = '.icns'
    icon_name = icon_root + icon_ext
    return os.path.join(app_path, 'Contents', 'Resources', icon_name)

# .. Basics ....................................................................

# Uncomment to override the output filename
# filename = 'test.dmg'

# Uncomment to override the output volume name
# volume_name = 'Test'

# Volume format (see hdiutil create -help)
format = defines.get('format', 'UDBZ')

# Volume size (must be large enough for your files)
size = defines.get('size', '100M')

# Files to include
files = [ application ]

# Symlinks to create
symlinks = { 'Applications': '/Applications' }

# Volume icon
#
# You can either define icon, in which case that icon file will be copied to the
# image, *or* you can define badge_icon, in which case the icon file you specify
# will be used to badge the system's Removable Disk icon
#
#icon = '/path/to/icon.icns'
badge_icon = icon_from_app(application)

# Where to put the icons
icon_locations = {
    appname:        (140, 120),
```

```
    'Applications': (500, 120)
    }

# .. Window configuration ..................................................

# Background
#
# This is a STRING containing any of the following:
#
#    #3344ff          - web-style RGB color
#    #34f             - web-style RGB color, short form (#34f == #3344ff)
#    rgb(1,0,0)       - RGB color, each value is between 0 and 1
#    hsl(120,1,.5)    - HSL (hue saturation lightness) color
#    hwb(300,0,0)     - HWB (hue whiteness blackness) color
#    cmyk(0,1,0,0)    - CMYK color
#    goldenrod        - X11/SVG named color
#    builtin-arrow    - A simple built-in background with a blue arrow
#    /foo/bar/baz.png - The path to an image file
#
# The hue component in hsl() and hwb() may include a unit; it defaults to
# degrees ('deg'), but also supports radians ('rad') and gradians ('grad'
# or 'gon').
#
# Other color components may be expressed either in the range 0 to 1, or
# as percentages (e.g. 60% is equivalent to 0.6).
background = 'builtin-arrow'

show_status_bar = False
show_tab_view = False
show_toolbar = False
show_pathbar = False
show_sidebar = False
sidebar_width = 180

# Window position in ((x, y), (w, h)) format
window_rect = ((100, 100), (640, 280))

# Select the default view; must be one of
#
#    'icon-view'
#    'list-view'
#    'column-view'
#    'coverflow'
#
default_view = 'icon-view'

# General view configuration
show_icon_preview = False

# Set these to True to force inclusion of icon/list view settings (otherwise
# we only include settings for the default view)
include_icon_view_settings = 'auto'
include_list_view_settings = 'auto'

# .. Icon view configuration ...............................................

arrange_by = None
grid_offset = (0, 0)
```

```
grid_spacing = 120
scroll_position = (0, 0)
label_pos = 'bottom' # or 'right'
text_size = 16
icon_size = 128

# .. List view configuration .................................................

# Column names are as follows:
#
#   name
#   date-modified
#   date-created
#   date-added
#   date-last-opened
#   size
#   kind
#   label
#   version
#   comments
#
list_icon_size = 16
list_text_size = 12
list_scroll_position = (0, 0)
list_sort_by = 'name'
list_use_relative_dates = True
list_calculate_all_sizes = False,
list_columns = ('name', 'date-modified', 'size', 'kind', 'date-added')
list_column_widths = {
    'name': 300,
    'date-modified': 181,
    'date-created': 181,
    'date-added': 181,
    'date-last-opened': 181,
    'size': 97,
    'kind': 115,
    'label': 100,
    'version': 75,
    'comments': 300,
    }
list_column_sort_directions = {
    'name': 'ascending',
    'date-modified': 'descending',
    'date-created': 'descending',
    'date-added': 'descending',
    'date-last-opened': 'descending',
    'size': 'descending',
    'kind': 'ascending',
    'label': 'ascending',
    'version': 'ascending',
    'comments': 'ascending',
    }
```

CHAPTER 2

Indices and tables

- *genindex*
- *modindex*
- *search*