ds_store Documentation

Release 1.0.0

Alastair Houghton

February 17, 2014

Contents

This document refers to version 1.0.0

CHAPTER 1

What is this?

Historically the Mac OS Finder stored additional per-file information in a special Finder Info field in the HFS/HFS+ filesystem. It also held other information in a single file known as the Desktop Database.

Filesystems other than HFS obviously do not have the Finder Info structure, and until recently support for extended attributes was rare. As a result, the Mac OS X Finder was written to store the necessary information in hidden files named `.DS_Store`, which it places into every directory where it needs to store information.

The format of these files is, sadly, not documented by Apple. This is a pain for software developers, who often distribute their software in Apple Disk Image (or `.dmg`) files. Typically developers set an attractive background on their disk images, increase the icon size and font size and often include a link to the `/Applications` folder. Unfortunately, the only supported way to set many of these things is via Finder itself. You might think that you could drive Finder with AppleScript for this purpose, but this turns out to be unreliable (Finder may not save the changes to the `.DS_Store` file immediately), and worse still Apple has made changes to the information Finder uses between versions of Mac OS X, such that setting some of these things on newer versions of the OS X Finder will not set them for users of older versions.

This module allows programmatic access to and construction of `.DS_Store` files directly from Python, with no Mac OS X specific code involved.

CHAPTER 2

Usage

Typical usage looks like this:

```
from ds_store import DSStore

with DSStore.open('/Users/alastair/.DS_Store', 'r+') as d:
  # Position the icon for "foo.txt" at (128, 128)
  d['foo.txt']['Iloc'] = (128, 128)

  # Display the plists for this folder
  print d['.']['bwsp']
  print d['.']['icvp']
```

Importantly, *deleting the* `DSStore` *object is not sufficient to flush changes to disk*. If you use the `with` syntax above, changes you make to the `.DS_Store` file will automatically be persisted. Otherwise, you will need to call `flush()` or `close()` to flush your changes to disk.

Note that Finder generally places information about folders in the *containing* folder. The exception is that if it cannot write to the containing folder, or the folder in question is at the root of a volume, Finder will put the information in a record for "." inside the folder to which it applies.

`ds_store` currently knows how to decode the following items

Table 2.1: Supported item codes

Code	Type	Python representation
`Iloc`	`blob`	`(x, y)` tuple
`bwsp`	`blob`	Property list (`dict`)
`lsvp`	`blob`	Property list (`dict`)
`lsvP`	`blob`	Property list (`dict`)
`icvp`	`blob`	Property list (`dict`)

Items not in the list above will be returned as `(type, value)` tuples. Supported `type` values are

Table 2.2: Suported type codes

Type	Python representation
`bool`	Boolean (`True` or `False`)
`long`	Integer
`shor`	Integer
`ustr`	Unicode string
`type`	4-character byte string
`comp`	Integer
`dutc`	Integer
`blob`	Byte string

If `ds_store` happens across any other type code, it will raise `ValueError`. This is unavoidable because the `.DS_Store` file format does not include length information, so if we find a type code we do not support, we cannot read the file.

CHAPTER 3

Code Documentation

3.1 ds_store package

class ds_store.**DSStore**(*store*)

Bases: object

Python interface to a .DS_Store file. Works by manipulating the file on the disk—so this code will work with .DS_Store files for *very* large directories.

A DSStore object can be used as if it was a mapping, e.g.:

```
d['foobar.dat']['Iloc']
```

will fetch the "Iloc" record for "foobar.dat", or raise KeyError if there is no such record. If used in this manner, the DSStore object will return (type, value) tuples, unless the type is "blob" and the module knows how to decode it.

Currently, we know how to decode "Iloc", "bwsp", "lsvp", "lsvP" and "icvp" blobs. "Iloc" decodes to an (x, y) tuple, while the others are all decoded using biplist.

Assignment also works, e.g.:

```
d['foobar.dat']['note'] = ('ustr', u'Hello World!')
```

as does deletion with del:

```
del d['foobar.dat']['note']
```

This is usually going to be the most convenient interface, though occasionally (for instance when creating a new .DS_Store file) you may wish to drop down to using DSStoreEntry objects directly.

class **Partial**(*store*, *filename*)

Bases: object

This is used to implement indexing.

DSStore.**close**()

Flush dirty data and close the underlying file.

DSStore.**delete**(*filename*, *code*)

Delete an item, identified by filename and code from the B-Tree.

DSStore.**find**(*filename*, *code=None*)

Returns a generator that will iterate over matching entries in the B-Tree.

DSStore.**flush**()
Flush any dirty data back to the file.

DSStore.**insert**(*entry*)
Insert `entry` (which should be a `DSStoreEntry`) into the B-Tree.

classmethod DSStore.**open**(*file_or_name*, *mode=u'r+'*, *initial_entries=None*)
Open a `.DS_Store` file; pass either a Python file object, or a filename in the `file_or_name` argument and a file access mode in the `mode` argument. If you are creating a new file using the "w" or "w+" modes, you may also specify a list of entries with which to initialise the file.

class ds_store.**DSStoreEntry**(*filename*, *code*, *typecode*, *value=None*)
Bases: `object`

Holds the data from an entry in a `.DS_Store` file. Note that this is not meant to represent the entry itself—i.e. if you change the type or value, your changes will *not* be reflected in the underlying file.

If you want to make a change, you should either use the `DSStore` object's `DSStore.insert()` method (which will replace a key if it already exists), or the mapping access mode for `DSStore` (often simpler anyway).

byte_length()
Compute the length of this entry, in bytes

classmethod read(*block*)
Read a `.DS_Store` entry from the containing Block

write(*block*, *insert=False*)
Write this entry to the specified Block

CHAPTER 4

Indices and tables

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

d