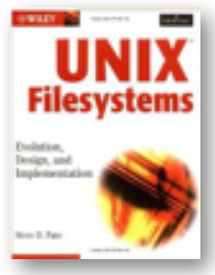

Panorama X
Developer Certification Training
Copyright © 2016 ProVUE Development

Files, Paths & Directories

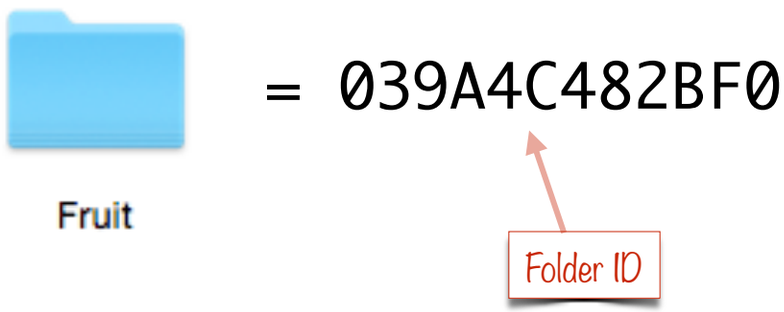
HFS vs. UNIX



Remember the first day when I talked about Panorama 6 being written using Carbon, while Panorama X uses Cocoa? This has a big effect on files and directories. Carbon uses HFS, which from a programmer's perspective hasn't changed since around 1985-86. Cocoa uses UNIX, which is quite different.

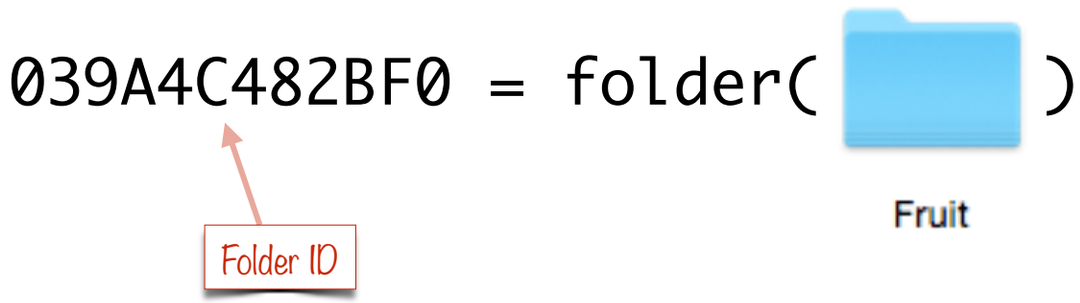
For Panorama X, the goal was to keep compatibility with your existing programs that were written for HFS, but also to modernize everything so that new applications can be written using the UNIX filesystem. This was pretty tricky, but I managed to do it. Let me show you how.

Folder ID



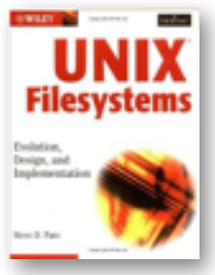
HFS has a unique 6 byte binary identifier for every folder.
This is called a "Folder ID"
There are many Panorama statements and functions that take a folder ID as a parameter, your programs are full of them.

Folder ID



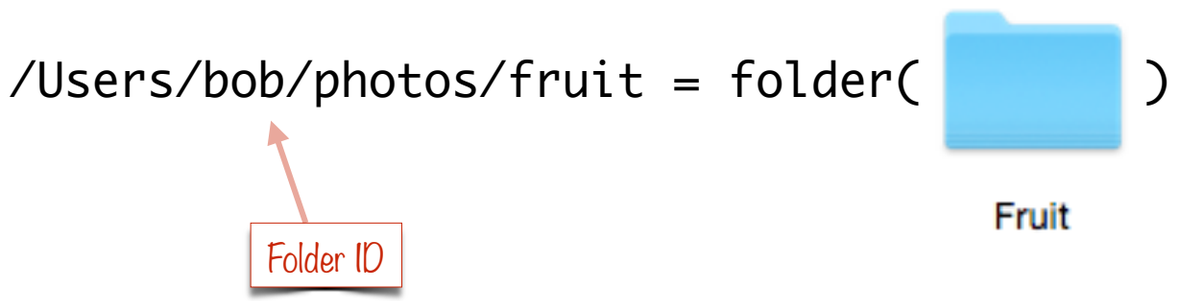
In Panorama 6, folder ID's come from the folder(function.
You would then pass the folder id on to other functions like fileload(, listfiles(, etc.
You never access the contents of the folder ID directly, no one ever looks at them or directly manipulates them.

Folder ID?



In UNIX, there is no such thing as a folder id.
So, what do do?

Folder ID



In Panorama X, a folder id is no longer a mysterious binary value.
Instead, it is simply the UNIX path of the folder -- with no / on the end.
You can still use the folder(function to create a folder ID.
Or, you can just type in the path -- as long as you are careful.
The folder(function does check to make sure that the specified folder does exist, and returns "" if it doesn't (same as Panorama 6).
DEMO - dbinfo("folder","") in Panorama 6 and Panorama X

Folder ID

```
fileload(folder("HD:Users:Bob:Documents:"),"Data.txt")
```

The bottom line is that your existing code that uses the folder(function will still work just fine.

Folder ID

```
fileload("/Users/Bob/Documents","Data.txt")
```

You can also just type in the UNIX path for the folder ID, like this.
It must be a complete path, not a partial path.

Folder ID

```
fileload("HD:Users:Bob:Documents:","Data.txt")
```

But this will NOT work with HFS paths -- the folder ID must be a UNIX path.

This is not really a problem, because as the next slide shows, in new code you can skip folder id's entirely!

Skip Folder ID Entirely!

```
fileload("/Users/Bob/Documents/Data.txt")
```

Most statements and functions that used to require separate parameters for folder id and filename can now accept a single parameter that combines both the path and the filename!

Skip Folder ID Entirely!

```
fileload("HD:Users:Bob:Documents:Data.txt")
```

This will also work with HFS paths!
If the path contains colons, Panorama X treats it as an HFS path.
Otherwise it treats it as a unix path.
If you mix colons and slashes, it will treat it as an HFS path, but it may get confused. So don't do that.

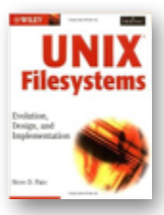
Skip Folder ID Entirely!

filedate(
fileexists(
fileinfo(
fileload(
fileloadpartial(
filesize(
filesuperdate(
filetime(
filetypecreator(
folderexists(

fileerase
filesave
fileappend
filerename
openanything
revealinfinder

These 10 functions and 6 statements all allow either 2 parameter (folder id and filename) or 1 parameter (filepath).
There may be others.
For new code I generally recommend forgetting about Folder ID's and just using paths.
If you happen to be listening to this and have never used Panorama 6, forget that you ever heard of Folder ID's even though Panorama X supports them -- you don't need them ever.

HFS vs. UNIX Full Paths



HD:Users:Bob:Documents:Data.txt

disk drive name

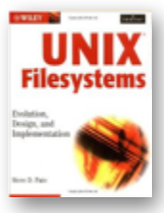
folder names

filename

Here's the format of a full HFS file path.
It *always* starts with the disk drive name.
Each item is separated by a colon. Colons aren't allowed in filenames or folder names.
This is the only kind of path that Panorama 6 understood.

HFS vs. UNIX Full Paths

`HD:Users:Bob:Documents:Data.txt`



`/Users/Bob/Documents/Data.txt`

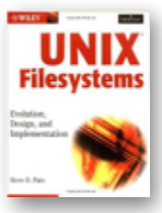
folder names

filename

Here's the format of a full UNIX file path *for a file that is on the internal hard drive.*
The disk drive name isn't mentioned at all!
Each item is separated by a slash. Slashes aren't allowed in filenames or folder names.

HFS vs. UNIX Full Paths

HD:Users:Bob:Documents:Data.txt



/Users/Bob/Documents/Data.txt

/Volumes/HD/Documents/Fruit/Data.txt

disk drive name

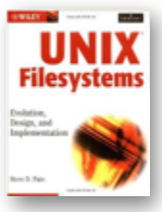
folder names

filename

Here's the format of a full UNIX file path *for a file that is on an external hard drive*.
always starts with /Volumes/, followed by the drive name.

HFS vs. UNIX Full Paths

HD:Users:Bob:Documents:Data.txt



/Users/Bob/Documents/Data.txt

/Volumes/HD/Documents/Fruit/Data.txt

Panorama X accepts all of these path formats.

It automatically figures out which one you have provided.

I recommend using UNIX paths going forward, but your existing programs that use HFS will still work

HFS vs. UNIX Full Paths

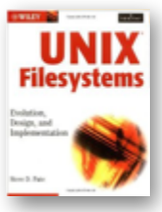
```
hfspath = hfspath(unixpath)

unixpath = unixpath(hfspath)
```

You'll probably never need to convert between HFS and UNIX path formats, but if you do, here are the functions for doing that.

HFS vs. UNIX Relative Paths

`Subfolder:Data.txt`

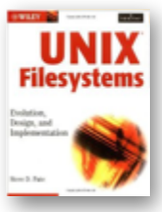


`Subfolder/Data.txt`

If a file you need to access is in a subfolder of the folder containing the currently active database, you can specify just a partial path.
This partial path should never begin with a colon or slash.
If the file is in the same folder as the current database, you can just use the filename by itself.

HFS vs. UNIX Name Lengths

31 Characters



255 Characters

FYI - Panorama X has no limit on filename or foldername length, it no longer chokes on names > 31 characters.
However, OS X does have a 255 character limit.

File Name vs. Display Name

```
filedisplayname(path)
```

OS X allows a file to have a "display name" that is different than the actual name.
You will rarely encounter this, but some special system items do this.
For example I think some special system items have display names that are localized, while the actual name is in English.
Panorama normally deals in the actual file name, but you can find out the display name of any file using the filedisplayname(function.
If you try this, usually you'll find that for most files the filename and display name are the same.

File Type Codes & Extensions

Type Code	Creator	Extensions	Notes
ZEPD	KASX	.pandb	Panorama X Database
KSET	KASX	.panset	Panorama X File Set
TEXT		.txt .html .css .csv	Text File
JPEG		.jpg .jpeg .jpe .jfif	JPEG Image File
PNGf		.png	PNG Image File
TIFF		.tiff .tif	TIFF Image File
GIFf		.gif	GIF Image File
PICT		.pict .pct .pic	PICT Image File (OS 9)
BMPf		.bmp	BMP Image File (Windows)
PDF		.pdf	PDF Document (not space on end of type code)
8BPS		.psd	Photoshop File
MOOV		.mov .moov .qt	QuickTime Movie File
WAVE		.wav	Audio File
MPEG		.mpg .mpeg .mp3 .mpe .mpa .mtv	MPEG Media File
AIFF		.aif .aiff	Audio File

On Mac OS 9 (and before), each file had a hidden 4 letter code called a type code. This hidden code identified the type of the file.
Cocoa does not use hidden type codes. Instead, Cocoa uses a visible code on the end of the filename. This visible code always starts with a period, and is called an extension.
Panorama X requires that files have a visible extension (don't be confused by the fact that the extension can be hidden in the Finder).

File Type Codes & Extensions

```
listfiles(path,"ZEPD")
```

now means "files that end with .pandb"

Your existing Panorama programs may use type codes for specifying what type of file to access.
This example tells the listfiles function to list all Panorama databases in the specified folder.
This code will still work, even though the database files don't actually have a hidden type code.
Panorama automatically figures out that ZEPD means "files that end with .pandb"

notiece that Panorama X will NOT find the file if it happens to actually have a hidden type code of ZEPD
Panorama X ignores hidden type codes, it only goes by the visible file extension.

File Type Codes & Extensions

```
listfiles(path,"ZEPD")

listfiles(path,".pandb")
```

For new code, you can now specify using the actual extension, or you can keep using the type code.

File Type Codes & Extensions

```
listfiles(path,"JPEGPNGfGIFf")

listfiles(path,".jpg.jpeg.png.gif")
```

Most place where you can specify a single type code, you can also specify multiple type codes.
Multiple extension are also allowed, just pack them in one after the other (no spaces)
These two examples lists all kinds of image files.

File Dialogs

Statement	Notes
OpenFileDialog	Same as Panorama 6
SaveFileDialog	Same as Panorama 6
ChooseFileDialog	New!
SaveDialog	New!

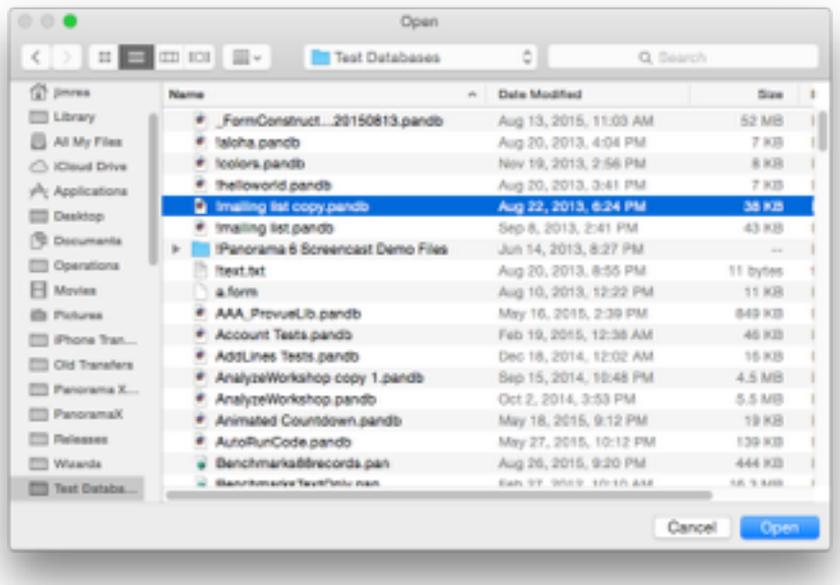
Sometimes you need to select a file with a dialog, or specify a name and location for a file you want to save.
The two older statements, OpenFileDialog and SaveFileDialog, work just as they did before, so that they are compatible with your existing databases.
There are two all new statements that give you MUCH more flexibility and control.
In fact, these new statements are so powerful, the older statements were actually implemented via the new statements.

OpenFileDialog vs. ChooseFileDialog

	OpenFileDialog	ChooseFileDialog
Number of files that can be chosen	One	Multiple
Result is returned as	FolderID,Name	Path
Specify initial folder	No	Yes
Custom dialog title	No	Yes
Select folders?	No	Yes
Custom buttons?	No	Yes
Show hidden files?	No	Optional
Open packages	No	Optional

This table shows at a glance the advantages of the ChooseFileDialog statement.
For simple applications, ChooseFileDialog is also simpler to use.

OpenFileDialog vs. ChooseFileDialog



Just to make sure we're all on the same page, here is the dialog we are talking about -- a standard system file selection dialog.

OpenFileDialog vs. ChooseFileDialog

```
local file,folder,type
openfiledialog folder,file,type,""
```

Here is the simplest possible code for picking a file with openfiledialog.

OpenFileDialog vs. ChooseFileDialog

```
local file,folder,type
openfiledialog folder,file,type,""
```

```
local filepath
choosefiledialog filepath
```

Here's the same thing using the ChooseFileDialog statement.
As you can see, it's a bit simpler.
And instead of returning a separate folder ID and filename, it's just a combined path and filename

OpenFileDialog vs. ChooseFileDialog

```
local file,folder,type
openfiledialog folder,file,type,"TEXT"
if file="" rtn endif /* in case cancelled */

openfile "+"+folderpath(folder)+file
```

```
local filepath
choosefiledialog filepath,".txt.html.csv"
if filepath="" rtn endif /* in case cancelled */

openfile "+"+filepath
```

Here is a real world example.
The dialog is used to pick a text file.
Then that text file is appended to the current database.

OpenFileDialog vs. ChooseFileDialog

```
local file,folder,type
openfiledialog folder,file,type,"TEXT"
if file="" rtn endif /* in case cancelled */

openfile "+"+folderpath(folder)+file
```

```
local filepath
choosefiledialog filepath,".txt.html.csv","~/Documents"
if filepath="" rtn endif /* in case cancelled */

openfile "+"+filepath
```

This third parameter tells Panorama to always start the dialog at the Documents folder.

ChooseFileDialog Advanced Options

```
choosefiledialog filepath,"Option1","Value1","Option2","Value2","Option3","Value3"
```

To use the advanced options of this statement, you specify one or more pairs of options and values

ChooseFileDialog Advanced Options

```
choosefiledialog filepath,"Option1","Value1","Option2","Value2","Option3","Value3"

choosefiledialog filepath,
    "filetypes",".jpg.jpeg.png.tif.tiff",
    "initialpath","~/Pictures",
    "multiple","true",
    "title","Select one or more images",
    "button","Select"
```

Here is an example with 5 pairs of options and values.
I've split each option onto a separate line for clarity, but this isn't necessary.
Note that these options can occur in any order, and you can leave out any options that you don't need.

ChooseFileDialog Advanced Options

```
choosefiledialog filepath,
    "filetypes",".jpg.jpeg.png.tif.tiff",
    "initialpath","~/Pictures",
    "multiple","true",
    "title","Select one or more images",
    "button","Select"
```

Let's take a look at each of the available options.

First is filetypes. Leave this option off if you want to allow any type of file (or use "").

Otherwise, list each extension you want to allow. No spaces.

You can also use 4 letter type codes (but you can't mix type codes and extensions in the same parameter).

```
local filepath
choosefiledialog filepath,
    "filetypes",".jpg.jpeg.png.tif.tiff",
    "initialpath","~/Pictures",
    "multiple","true",
    "title","Select one or more images",
    "button","Select"
displaydata filepath
```

ChooseFileDialog Advanced Options

```
choosefiledialog filepath,
    "filetypes",".jpg.jpeg.png.tif.tiff",
    "initialpath","~/Pictures",
    "multiple","true",
    "title","Select one or more images",
    "button","Select"
```

Initial path specifies what folder will be open when the dialog first opens (if you care).

```
local filepath
choosefiledialog filepath,
    "filetypes",".jpg.jpeg.png.tif.tiff",
    "initialpath","~/Pictures",
    "multiple","true",
    "title","Select one or more images",
    "button","Select"
displaydata filepath
```

ChooseFileDialog Advanced Options

```
choosefiledialog filepath,
    "filetypes",".jpg.jpeg.png.tif.tiff",
    "initialpath","~/Pictures",
    "multiple","true",
    "title","Select one or more images",
    "button","Select"
```

~/ is current user's home folder

By the way, if you don't know, ~/ is a shortcut for the current user's home folder.
This is an Apple thing, not a Panorama specific thing.

```
local filepath
choosefiledialog filepath,
    "filetypes",".jpg.jpeg.png.tif.tiff",
    "initialpath","~/Pictures",
    "multiple","true",
    "title","Select one or more images",
    "button","Select"
displaydata filepath
```

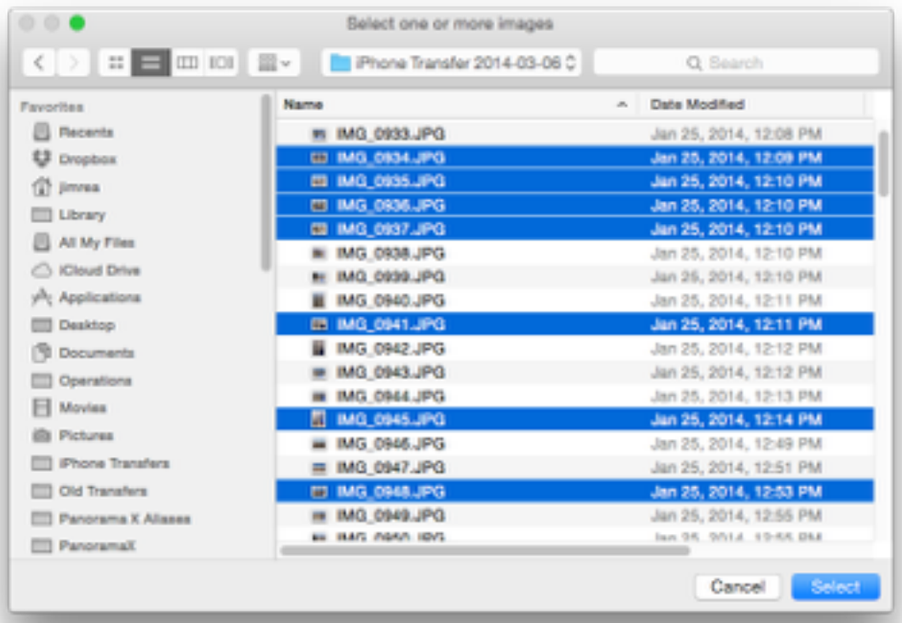
ChooseFileDialog Advanced Options

```
choosefiledialog filepath,
    "filetypes",".jpg.jpeg.png.tif.tiff",
    "initialpath","~/Pictures",
    "multiple","true",
    "title","Select one or more images",
    "button","Select"
```

Normally you can select only one file at at time.
But if multiple is true, you can select more than one.
DEMO

```
local filepath
choosefiledialog filepath,
    "filetypes",".jpg.jpeg.png.tif.tiff",
    "initialpath","~/Pictures",
    "multiple","true",
    "title","Select one or more images",
    "button","Select"
displaydata filepath
```

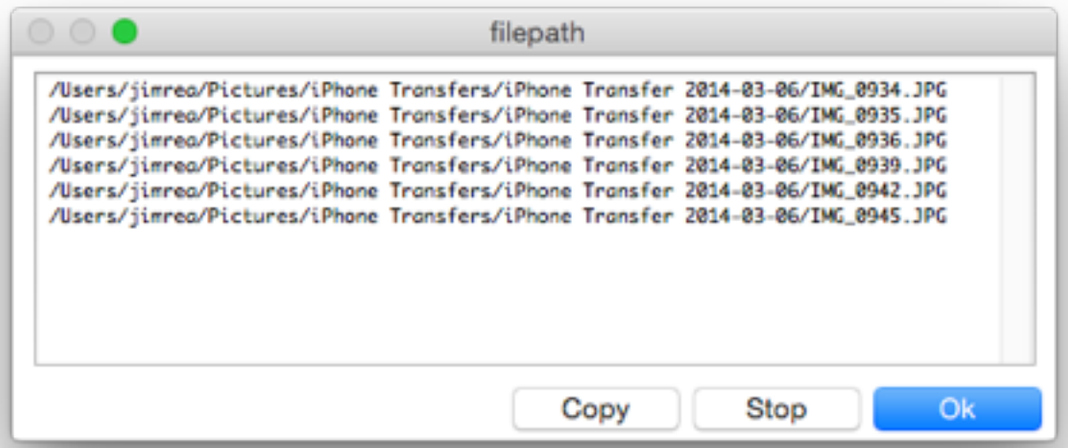
ChooseFileDialog Advanced Options



Hold down the shift key to select a bunch of files in a row.
Hold down the Command key to select individual files.

```
local filepath
choosefiledialog filepath,
    "filetypes",".jpg.jpeg.png.tif.tiff",
    "initialpath","~/Pictures",
    "multiple","true",
    "title","Select one or more images",
    "button","Select"
displaydata filepath
```

ChooseFileDialog Advanced Options

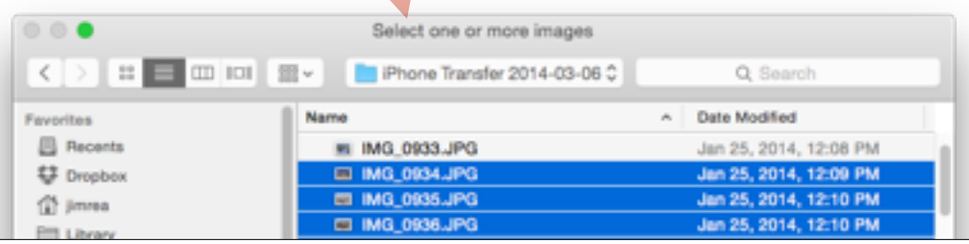


When the dialog is finished, you'll get a carriage return delimited list of the selected files, like this
Including the complete folder path for each selected file.

```
local filepath
choosefiledialog filepath,
    "filetypes",".jpg.jpeg.png.tif.tiff",
    "initialpath","~/Pictures",
    "multiple","true",
    "title","Select one or more images",
    "button","Select"
displaydata filepath
```

ChooseFileDialog Advanced Options

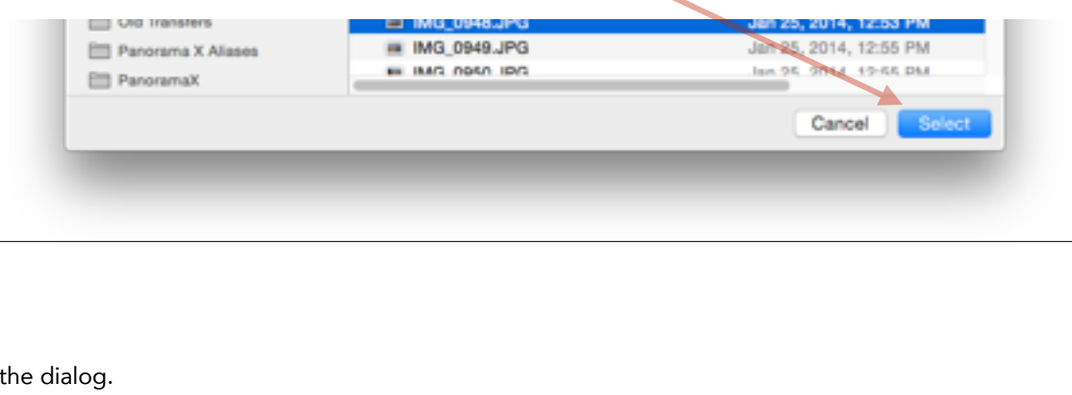
```
choosefiledialog filepath,
    "filetypes",".jpg.jpeg.png.tif.tiff",
    "initialpath","~/Pictures",
    "multiple","true",
    "title","Select one or more images",
    "button","Select"
```



The title option allows you to customize the title that appears at the top of the dialog.

ChooseFileDialog Advanced Options

```
choosefiledialog filepath,
    "filetypes",".jpg.jpeg.png.tif.tiff",
    "initialpath","~/Pictures",
    "multiple","true",
    "title","Select one or more images",
    "button","Select"
```



The button option allows you to customize the default button of the dialog.
You can only make this up to about three words, wider than that and it will get cut off.

ChooseFileDialog Advanced Options

```
choosefiledialog filepath,
    "folders","true",
    "files","false"
```

Normally you can't select folders, only files.
If you try to select a folder, it will just open the folder so that you can see the files inside.
If you set this option to true, you can select a folder, use this if you want to pick a folder instead of a file.
This is option used with the files option. Set this option if you don't want to allow files selected, only folders.
DEMO

```
local filepath
choosefiledialog filepath,
    "folders","true",
    "files","false"
message filepath
```

ChooseFileDialog Advanced Options

```
choosefiledialog filepath,
    "showhidden","true",
    "openpackages","true"
```

These are advanced options.
If showhidden is true, hidden files will be visible.
if openpackages is true, you can go inside packages and choose files inside the packages
        for example inside applications, or inside Panorama files.
DEMO

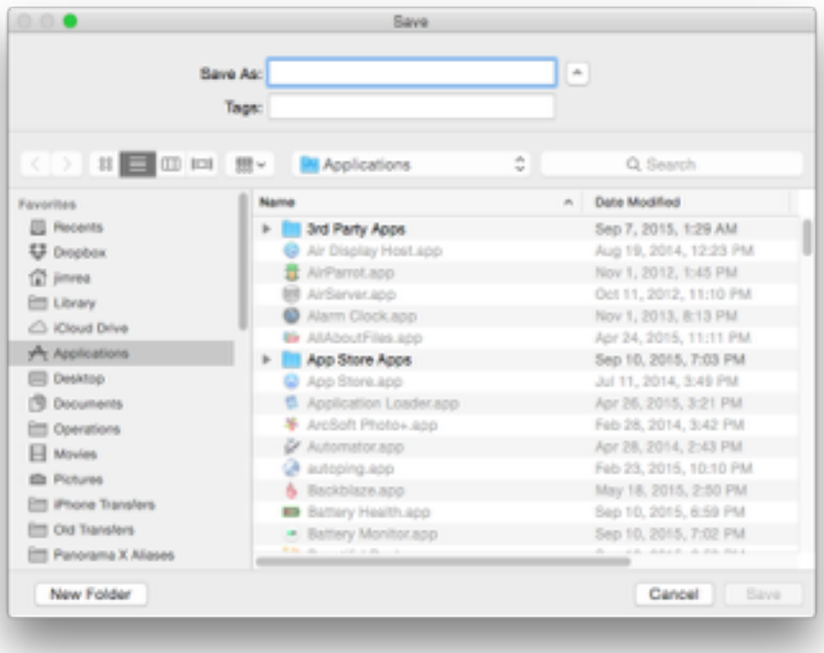
```
local filepath
choosefiledialog filepath,
   "showhidden","true",
   "openpackages","true"
message filepath
```

SaveFileDialog vs. SaveDialog

	SaveFileDialog	SaveDialog
Specify default file name	Yes	Yes
Result is returned as	FolderID,Name	Path
Specify initial folder	No	Yes
Custom dialog title?	No	Yes
Custom prompt?	Yes	Yes
Specify allowed file types?	No	Yes
Custom buttons?	No	Yes
Disable creating new folders?	No	Yes
Show hidden files?	No	Optional
Open packages	No	Optional

This table shows at a glance the advantages of the SaveDialog statement.
For simple applications, SaveDialog is also simpler to use.

SaveFileDialog vs. SaveDialog



Just to make sure we're all on the same page, here is the dialog we are talking about -- a standard system file save dialog.

```
local fileChoice
savedialog fileChoice
```

SaveDialog Statement

```
local fileChoice
savedialog fileChoice
```

Here is the simplest use of the savedialog statement.
It returns the chosen path and filename in fileChoice, or "" if cancel was pressed.

```
local fileChoice
savedialog fileChoice
```

SaveDialog Advanced Options

```
savedialog filepath,"Option1","Value1","Option2","Value2","Option3","Value3"
```

To use the advanced options of this statement, you specify one or more pairs of options and values

SaveDialog Advanced Options

```
savedialog filepath,"Option1","Value1","Option2","Value2","Option3","Value3"
```

```
savedialog filepath,
    "InitialPath","~/Pictures",
    "FileName","Daily Image "+datepattern(today(),"YYYY-MM-DD")+".jpg",
    "FileTypes",".jpg.png",
    "AllowAnyFileType","true",
    "Title","Save today's image",
    "Button","Save Image",
    "Prompt","Image Name",
    "CanCreateFolders","false"
```

Here is an example with 8 pairs of options and values.
I've split each option onto a separate line for clarity, but this isn't necessary.
Note that these options can occur in any order, and you can leave out any options that you don't need.

```
local filepath
savedialog filepath,
    "InitialPath","~/Pictures",
    "FileName","Daily Image "+datepattern(today(),"YYYY-MM-DD")+".jpg",
    "FileTypes",".jpg.png",
    "AllowAnyFileType","true",
    "Title","Save today's image",
    "Button","Save Image",
    "Prompt","Image Name:",
    "CanCreateFolders","false"
message filepath
```

SaveDialog Advanced Options

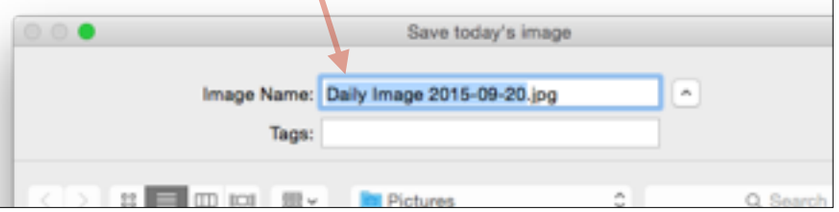
```
savedialog filepath,
    "InitialPath","~/Pictures",
    "FileName","Daily Image "+datepattern(today(),"YYYY-MM-DD")+".jpg",
    "FileTypes",".jpg.png",
    "AllowAnyFileType","true",
    "Title","Save today's image",
    "Button","Save Image",
    "Prompt","Image Name",
    "CanCreateFolders","false"
```

Once again, initial path specifies what folder will be open when the dialog first opens (if you care).

```
local filepath
savedialog filepath,
    "InitialPath","~/Pictures",
    "FileName","Daily Image "+datepattern(today(),"YYYY-MM-DD")+".jpg",
    "FileTypes",".jpg.png",
    "AllowAnyFileType","true",
    "Title","Save today's image",
    "Button","Save Image",
    "Prompt","Image Name:",
    "CanCreateFolders","false"
message filepath
```

SaveDialog Advanced Options

```
savedialog filepath,
    "InitialPath","~/Pictures",
    "FileName","Daily Image "+datepattern(today(),"YYYY-MM-DD")+".jpg",
    "FileTypes",".jpg.png",
    "AllowAnyFileType","true",
    "Title","Save today's image",
    "Button","Save Image",
    "Prompt","Image Name",
    "CanCreateFolders","false"
```

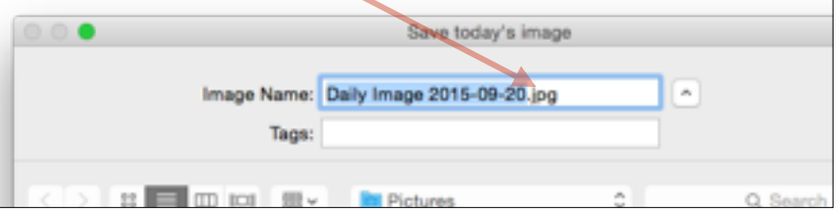


This specifies the default filename. If you want an empty default filename, just leave this blank.
This example will provide a default name with today's date -- *Daily Image 2015-09-22.jpg*

```
local filepath
savedialog filepath,
    "InitialPath","~/Pictures",
    "FileName","Daily Image "+datepattern(today(),"YYYY-MM-DD")+".jpg",
    "FileTypes",".jpg.png",
    "AllowAnyFileType","true",
    "Title","Save today's image",
    "Button","Save Image",
    "Prompt","Image Name:",
    "CanCreateFolders","false"
message filepath
```

SaveDialog Advanced Options

```
savedialog filepath,
    "InitialPath","~/Pictures",
    "FileName","Daily Image "+datepattern(today(),"YYYY-MM-DD")+".jpg",
    "FileTypes",".jpg.png",
    "AllowAnyFileType","true",
    "Title","Save today's image",
    "Button","Save Image",
    "Prompt","Image Name",
    "CanCreateFolders","false"
```



This specifies what filetype extension are allowed.
The first extension listed is the default extension
This will override any extension provided in the filename option.
If the user tries to type in an extension that is not on this list, the first extension listed will be appended to the name.
Note: You can also use 4 character type codes here.

```
local filepath
savedialog filepath,
    "InitialPath","~/Pictures",
    "FileName","Daily Image "+datepattern(today(),"YYYY-MM-DD")+".jpg",
    "FileTypes",".jpg.png",
    "AllowAnyFileType","true",
    "Title","Save today's image",
    "Button","Save Image",
    "Prompt","Image Name:",
    "CanCreateFolders","false"
message filepath
```

SaveDialog Advanced Options

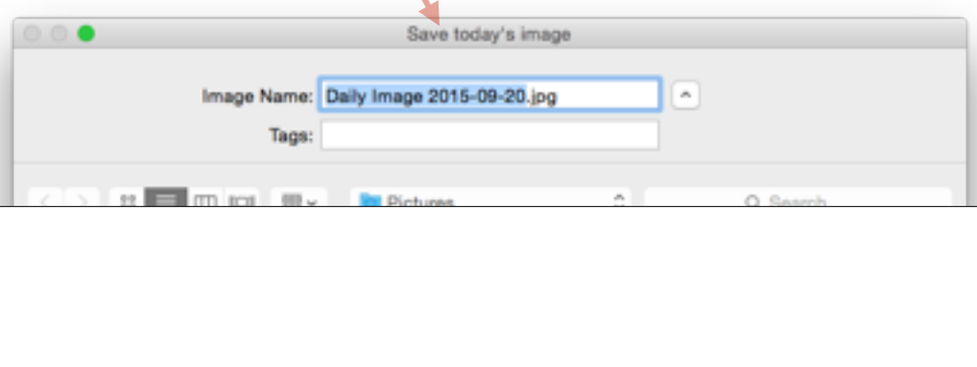
```
savedialog filepath,
    "InitialPath","~/Pictures",
    "FileName","Daily Image "+datepattern(today(),"YYYY-MM-DD")+".jpg",
    "FileTypes",".jpg.png",
    "AllowAnyFileType","true",
    "Title","Save today's image",
    "Button","Save Image",
    "Prompt","Image Name",
    "CanCreateFolders","false"
```

This modifies the filetypes option.
If allowanyfiletype is true, then the user can type in any extension they want.
However, if they type in an extension that is not on the list of filetypes, they will be warned.
    actually -- this parameter doesn't seem to have any effect???
So you should always list the filetype extensions you expect to be used.

```
local filepath
savedialog filepath,
    "InitialPath","~/Pictures",
    "FileName","Daily Image "+datepattern(today(),"YYYY-MM-DD")+".jpg",
    "FileTypes",".jpg.png",
    "AllowAnyFileType","true",
    "Title","Save today's image",
    "Button","Save Image",
    "Prompt","Image Name:",
    "CanCreateFolders","false"
message filepath
```

SaveDialog Advanced Options

```
savedialog filepath,
    "InitialPath","~/Pictures",
    "FileName","Daily Image "+datepattern(today(),"YYYY-MM-DD")+".jpg",
    "FileTypes",".jpg.png",
    "AllowAnyFileType","true",
    "Title","Save today's image",
    "Button","Save Image",
    "Prompt","Image Name",
    "CanCreateFolders","false"
```

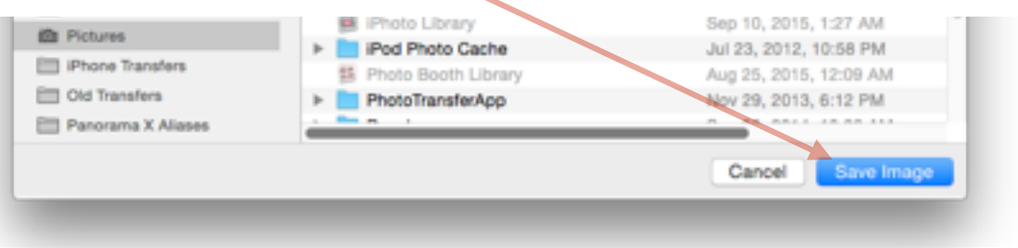


The title is displayed at the top of the dialog.

```
local filepath
savedialog filepath,
    "InitialPath","~/Pictures",
    "FileName","Daily Image "+datepattern(today(),"YYYY-MM-DD")+".jpg",
    "FileTypes",".jpg.png",
    "AllowAnyFileType","true",
    "Title","Save today's image",
    "Button","Save Image",
    "Prompt","Image Name:",
    "CanCreateFolders","false"
message filepath
```

SaveDialog Advanced Options

```
savedialog filepath,
    "InitialPath","~/Pictures",
    "FileName","Daily Image "+datepattern(today(),"YYYY-MM-DD")+".jpg",
    "FileTypes",".jpg.png",
    "AllowAnyFileType","true",
    "Title","Save today's image",
    "Button","Save Image",
    "Prompt","Image Name",
    "CanCreateFolders","false"
```

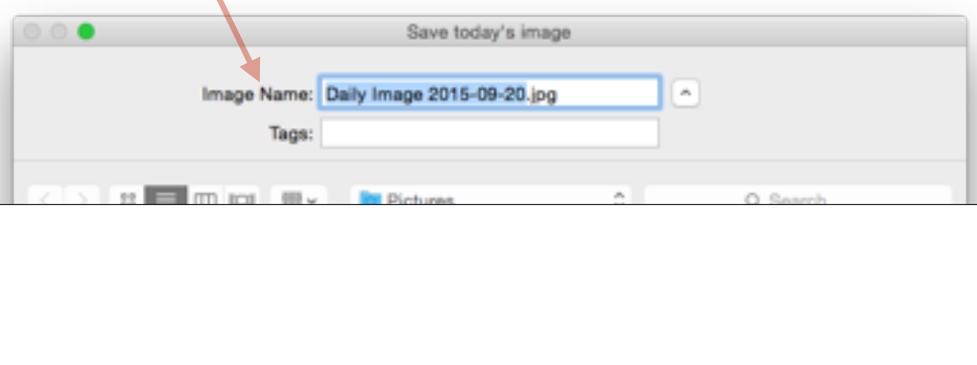


You can customize the button name, up to about 3 words.

```
local filepath
savedialog filepath,
    "InitialPath","~/Pictures",
    "FileName","Daily Image "+datepattern(today(),"YYYY-MM-DD")+".jpg",
    "FileTypes",".jpg.png",
    "AllowAnyFileType","true",
    "Title","Save today's image",
    "Button","Save Image",
    "Prompt","Image Name:",
    "CanCreateFolders","false"
message filepath
```

SaveDialog Advanced Options

```
savedialog filepath,
    "InitialPath","~/Pictures",
    "FileName","Daily Image "+datepattern(today(),"YYYY-MM-DD")+".jpg",
    "FileTypes",".jpg.png",
    "AllowAnyFileType","true",
    "Title","Save today's image",
    "Button","Save Image",
    "Prompt","Image Name",
    "CanCreateFolders","false"
```

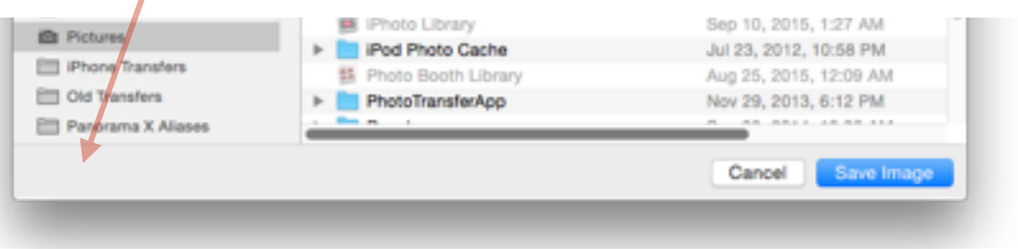


You can specify the prompt next to the filename.
It looks like you could put lots of text here, but OS X will cut off more than 2 or 3 words.

```
local filepath
savedialog filepath,
    "InitialPath","~/Pictures",
    "FileName","Daily Image "+datepattern(today(),"YYYY-MM-DD")+".jpg",
    "FileTypes",".jpg.png",
    "AllowAnyFileType","true",
    "Title","Save today's image",
    "Button","Save Image",
    "Prompt","Image Name:",
    "CanCreateFolders","false"
message filepath
```

SaveDialog Advanced Options

```
savedialog filepath,
    "InitialPath","~/Pictures",
    "FileName","Daily Image "+datepattern(today(),"YYYY-MM-DD")+".jpg",
    "FileTypes",".jpg.png",
    "AllowAnyFileType","true",
    "Title","Save today's image",
    "Button","Save Image",
    "Prompt","Image Name",
    "CanCreateFolders","false"
```

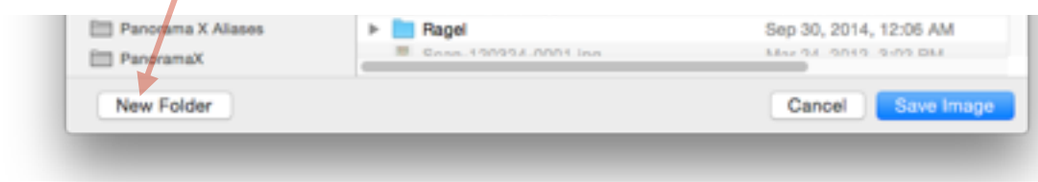


You can get rid of the new folder button.

```
local filepath
savedialog filepath,
    "InitialPath","~/Pictures",
    "FileName","Daily Image "+datepattern(today(),"YYYY-MM-DD")+".jpg",
    "FileTypes",".jpg.png",
    "AllowAnyFileType","true",
    "Title","Save today's image",
    "Button","Save Image",
    "Prompt","Image Name:",
    "CanCreateFolders","false"
message filepath
```

SaveDialog Advanced Options

```
savedialog filepath,
    "InitialPath","~/Pictures",
    "FileName","Daily Image "+datepattern(today(),"YYYY-MM-DD")+".jpg",
    "FileTypes",".jpg.png",
    "AllowAnyFileType","true",
    "Title","Save today's image",
    "Button","Save Image",
    "Prompt","Image Name",
    "CanCreateFolders","true"
```



if this option is true, or omitted entirely, the New Folder button appears in the bottom left of the dialog.

```
local filepath
savedialog filepath,
    "InitialPath","~/Pictures",
    "FileName","Daily Image "+datepattern(today(),"YYYY-MM-DD")+".jpg",
    "FileTypes",".jpg.png",
    "AllowAnyFileType","true",
    "Title","Save today's image",
    "Button","Save Image",
    "Prompt","Image Name:",
    "CanCreateFolders","false"
message filepath
```

SaveDialog Advanced Options

```
savedialog filepath,
    "showhidden","true",
    "openpackages","true"
```

These are advanced options.
If showhidden is true, hidden files will be visible.
if openpackages is true, you can go inside packages and save files inside the packages
for example inside applications, or inside Panorama files.


```
savedialog filepath,
    "showhidden","true",
    "openpackages","true"
```

Files, Paths & Directories