

brandxmp Group

John D. Baker

<https://github.com/bakerjd99/jackshacks/blob/main/brandxmp.ijs>

SHA-256: e1a8e960981b663b3d2709165997fbc38003b737d73d18139926cb09b21338ba

May 21, 2023

Contents

brandxmp Overview	2
brandxmp Interface	2
Installing brandxmp	2
Using brandxmp	3
brandxmp Source Code	5
=: Index	22

brandxmp Overview

brandxmp is a J script that brands sidecar [XMP](#) files generated by [Darktable](#).

brandxmp scans image directories containing XMP metadata files and inserts [Dublin Core](#) `title` elements like:

```
<dc:title>
  <rdf:Alt>
    <rdf:li xml:lang="x-default">_DSC8001.NEF|9a629837f0107ade4957352dfb55ecd483f932e2b46a63e6f56276a886c5611d</rdf:li>
  </rdf:Alt>
</dc:title>
```

The `title` string consists of the file name and the SHA256 hash of the corresponding image file. I never alter source image files so this hash will be a unique stable key associated with the original image.

brandxmp Interface

```
audbranddir [8] audit xmp/raw image directories
sidecars    [16] image raws with corresponding sidecar xmp files
titbranddir [18] brand eligible xmp files in directory
titbrandxmp [19] brand xmp sidecar file with file name and hash of associated image
```

Installing brandxmp

If you have a current version of J (9.0x+ or later) installed **brandxmp** can be downloaded as a J [addon](#) script by typing the following commands into a JQt or JHS session.

NB. install addon files in ~addons/jacks

```
install 'github:bakerjd99/jackshacks'
```

NB. installed files

```
dir '~addons/jacks'
```

NB. load script

```
load '~addons/jacks/brandxmp.ijs'
```

To update *brandxmp* and other files in *~addons/jacks* simply reinstall. New files will be added and existing files will be overwritten with current versions.

Using *brandxmp*

To use *brandxmp* do the following:

1. Set Darktable to check for altered XMP files. The default setting is to create XMP files and then rewrite them with only Darktable edits. You want to preserve XMP changes made outside of Darktable.
2. Add a directory filled with RAW images to the Darktable library. Darktable creates the XMP files *brandxmp* needs. Any image processing program that creates XMP files with Dublin Core elements can be used instead of Darktable.
3. Shutdown Darktable.
4. Start J, load *brandxmp*, and scan the directory of added RAW files:

```
load '~addons/jacks/brandxmp.ijs'  
titbranddir 'c:/your/raw/directory' NB. win path
```

5. Restart Darktable and merge the XMP changes to the Darktable library database.
6. It's also a good idea to run `audbranddir` over branded XMP directories. `audbranddir` writes a simple text file in the target directory that summarizes raws and retains the original file names and SHA256 hashes inserted in `title` elements.

```
audbranddir '/Users/your/pictures/to/be/audited' NB. macos/linux path
```

After merging the XMP changes Darktable will export “developed” RAW images with the branded `title` element. If you use other image editors make sure they maintain this element. My favorite editors [Picture Window Pro](#), [Affinity Photo](#), [The GIMP](#), [ThumbsPlus](#), and [macOS Photos](#) all maintain this element. Yours may not!

brandxmp Source Code

```
NB.*brandxmp s-- brand directories of xmp sidecar files with file name and hash.
NB.
NB. verbatim:
NB.
NB. interface word(s):
NB. -----
NB.  audbranddir - audit xmp/raw image directories
NB.  sidecars    - image raws with corresponding sidecar xmp files
NB.  titbranddir - brand eligible xmp files in directory
NB.  titbrandxmp - brand xmp sidecar file with file name and hash of associated image
NB.
NB. created: 2022jul13
NB. -----
NB. 22jul15 (audbrandxmp) added
NB. 22jul16 (audbrandxmp) renamed (audbranddir) to match (titbranddir)

coclass 'brandxmp'

NB.*dependents
NB. (*)=: shabrand wrecho RAWFILETYPES XMPBRDEL XMPTITLEFRAG
NB.*enddependents

NB. xmp brand delimiter character
XMPBRDEL=: ' | '
```

NB. brand file with name and sha256 hash: shabrand 'c:\temp\IMG_0162.jpg'

```
shabrand=: (XMPBRDEL ,~ justfileext@winpathsep) , sha256@read
```

NB. write bytes (x) and return file (y)

```
wrecho=: {{ y [ x (write :: _1:) y }}
```

NB. image types considered raw - adjust if necessary

```
RAWFILETYPES=: <;._1 ' jpg tif tiff nef dng png jpeg heic'
```

```
XMPTITLEFRAG=: (0 : 0)
```

```
<dc:title>
```

```
<rdf:Alt>
```

```
<rdf:li xml:lang="x-default">[~(fhash)~]</rdf:li>
```

```
</rdf:Alt>
```

```
</dc:title>
```

```
)
```

*NB.*end-header*

NB. carriage return character

```
CR=: 13{a.
```

NB. interface words (IFACEWORDSbrandxmp) group

```
IFACEWORDSbrandxmp=: <;._1 ' audbranddir sidecars titbranddir titbrandxmp'
```

NB. line feed character

```
LF=: 10{a.
```

NB. root words (ROOTWORDSbrandxmp) group

ROOTWORDSbrandxmp=: <;._1 ' IFACEWORDSbrandxmp ROOTWORDSbrandxmp VMDbrandxmp audbranddir titbranddir'

NB. version, make count and date

VMDbrandxmp=: '0.6.5';4;'21 May 2023 12:08:31'

NB. name and extension of xmp audit file

XMPAUDITFILE=: '00auditxmp.txt'

NB. xmp list line length

XMPWID=: 75

NB. retains string (y) after last occurrence of (x)

afterlaststr=:] }~ #@[+ 1&(i~)@([E.])

NB. retains string after first occurrence of (x)

afterstr=:] }~ #@[+ 1&(i~)@([E.])

allraws=: 3 : 0

*NB.*allraws v-- list all raw image files in directory.*

NB.

NB. monad: blcl =. allraws clDirectory

NB.

NB. r0=. 'c:\pictures\2022\North Rim Monument Valley\06_jun\d7500'

NB. r1=. '/Users/johnbaker/Pictures/2022/Idaho/07_jul/d7500'

```
NB.    allraws r0
NB.
NB. dyad:  blcl =. blclRawExt allraws clDirectory
NB.
NB.    (;'nef dng') allraws rp

RAWFILETYPES allraws y
:
NB. j profile !(*)=. dir IFUNIX

NB. file extension case matters on unix systems (sigh)
if. IFUNIX do. ext=. x,toupper&.> x else. ext=. x end.

a: -.~ , ;1&dir&.> (<(tslash2 y) , '*' ) ,&.> ext
)

NB. trims all leading and trailing blanks
alltrim=: ] #~ [: -. [: (*./\ . +. */.\) ' ' &=

audbranddir=: 3 : 0

NB.*audbranddir v-- audit xmp/raw image directories.
NB.
NB. Scans a directory containing raw image files where files with
NB. extensions from (RAWFILETYPES) are considered raw and writes
NB. a simple text file in the directory that lists:
NB.
```



```
NB. verbatim:
NB.
NB. 1. counts of raw types in directory
NB. 2. raws without sidecar xmps
NB. 3. raws with xmps missing titles
NB. 4. titles of xmps - shows name/hash if branded
NB.
NB. monad: clAuditFile =. audbranddir clDirectory
NB.
NB.  r0=. 'c:/pictures/2022/Washington/04_apr/d7500'
NB.  r1=. 'c:\pictures\2022\North Rim Monument Valley\06_jun\d7500'
NB.  r2=. '/Users/johnbaker/Pictures/2022/Idaho/07_jul/d7500'
NB.  audbranddir r0

NB. if no raws return empty and do not write audit file
if. 0 = #raw=. allraws y do. '' return. end.

NB. start file text
txt=. (timestamp ''),LF,y
txt=. txt,LF,LF,'raw file counts',LF,ctl extfreq raw

NB. raws without xmps
noxmp=. raw -. 0 {"1 sid=. sidecars y
txt=. txt,LF,LF,(":#noxmp),' raws without xmps'
if. #noxmp do.
  txt=. txt,LF,ctl XMPWID list '/'&afterlaststr.> noxmp
end.
```

NB. extract title element text from all xmps

```
elt=. 'dc:title'&geteletext@read.> 1 {"1 sid  
bm=. 0 = #&> elt
```

NB. xmps without title elements

```
ubxmp=. bm # 0 {"1 sid  
txt=. txt,LF,LF,(":#ubxmp),' rows with xmps missing titles'  
if. #ubxmp do.  
    txt=. txt,LF,ctl XMPWID list '/'&afterlaststr.> ubxmp  
end.
```

NB. xmps with titles

```
htxmp=. ;elt #~ -.bm  
txt=. txt,LF,LF,(":#htxmp),' rows with xmps having titles'  
if. #htxmp do.  
    NB. raw file names  
    rit=. ,&XMPBRDEL.> '/'&afterlaststr.> (0 {"1 sid) #~ -.bm  
    NB. title text  
    brtxt=. ("x-default">'&afterstr)@('</rdf:li>'&beforestr)&.> htxmp  
    txt=. txt,LF,ctl ;"1 rit ,. brtxt  
end.
```

NB. write audit file

```
afile [ (toHOST tlf txt) write afile=. (tslash2 y),XMPAUDITFILE  
)
```

NB. retains string before first occurrence of (x)

```
beforestr=: ] {~ 1&(i.~)@([ E. ])
```

```
betweenstrs=: 4 : 0
```

```
NB.*betweenstrs v-- select sublists between nonnested delimiters
```

```
NB. discarding delimiters.
```

```
NB.
```

```
NB. dyad: blcl =. (clStart;clEnd) betweenstrs cl
```

```
NB.      blnl =. (nlStart;nlEnd) betweenstrs nl
```

```
NB.
```

```
NB. ('start';'end') betweenstrs 'start yada yada end boo hoo start ahh end'
```

```
NB.
```

```
NB. NB. also applies to numeric delimiters
```

```
NB. (1 1;2 2) betweenstrs 1 1 66 666 2 2 7 87 1 1 0 2 2
```

```
's e'=. x
```

```
llst=. ((-#s) (|.!.0) s E. y) +. e E. y
```

```
mask=. ~:/\ llst
```

```
(mask#llst) <|.1 mask#y
```

```
)
```

```
NB. boxes open nouns
```

```
boxopen=: <^(L. = 0:)
```

```
changestr=: 4 : 0
```

```
NB.*changestr v-- replaces substrings - see long documentation.
```

```

NB.
NB. dyad:  clReps changestr cl
NB.
NB.  NB. first character delimits replacements
NB.  '/change/becomes/me/ehh' changestr 'blah blah ...'

pairs=. 2 {."(1) _2 [\ <; _1 x      NB. change table
cnt=. _1 [ lim=. # pairs
while. lim > cnt=.>:cnt do.        NB. process each change pair
  't c'=. cnt { pairs              NB. /target/change
  if. +./b=. t E. y do.            NB. next if no target
    r=. I. b                       NB. target starts
    'l q'=. #&> cnt { pairs         NB. lengths
    p=. r + 0,+/\(<:# r)$ d=. q - 1 NB. change starts
    s=. * d                         NB. reduce < and > to =
    if. s = _1 do.
      b=. 1 #~ # b
      b=. ((1 * # r)$ 1 0 #~ q,l-q) (,r +/ i. l)} b
      y=. b # y
      if. q = 0 do. continue. end.  NB. next for deletions
    elseif. s = 1 do.
      y=. y #~ >: d r} b            NB. first target char replicated
    end.
    y=. (c $~ q *# r) (,p +/i. q)} y NB. insert replacements
  end.
end. y                             NB. altered string
)

```

```

NB. character table to newline delimited list
ctl=: }.@(@1&(",1)@(-.@(*./\"1@(&' ' @])))) # ,@((10{a.)&(",1)@]))

cutnestidx=: 4 : 0

NB.*cutnestidx v-- cut list into nested runs and other.
NB.
NB. Nested runs are delimited by begin and end tags. This verb is
NB. oriented toward XML parsing where typical begin end tags are
NB. <ul> </ul> and tags with attributes like: <hoo boy="2">
NB. </hoo>
NB.
NB. This verb can process numeric lists but care must be taken to
NB. insure the pad item (1{.0$y) does not match begin and end
NB. values.
NB.
NB. dyad: (ilIdx ;< blcl) =. (clStart;clEnd) cutnestidx cl
NB.       (ilIdx ;< blnl) =. (nlStart;nlEnd) cutnestidx nl
NB.
NB. xml=. 'yada <ol><li>one</li><ol><li>sub one</li></ol></ol> boo'
NB. ('<ol>';'</ol>') cutnestidx xml
NB.
NB. 88 99 cutnestidx (i.5),88,(10?10),99 88 5 5 5 5 5 99

if. #y do.
  's e'=. ,&.> x          NB. start end lists
  ut=. 1{.0$y             NB. padding
  assert. -.s -: e        NB. they must differ

```

```

assert. -(s -:ut) +. e -:ut
sp=. s E. ut=.y,ut          NB. start mask

NB. quit if no delimiters
if. -.1 e. sp do. (i.0);<<y return. end.

ep=. e E. ut                NB. end mask
assert. (+/sp) = +/ep        NB. basic balance
dp=. sp + - ep              NB. start end marks
assert. 0 *./ . <: +/\ dp    NB. nested balance
ep=. I. _1=dp [ sp=. I. 1=dp  NB. start end indexes
ut=. +/\dp -. 0              NB. scanned marks
dp=. /:~ sp,ep               NB. all indexes
sp=. (firstones 1<:ut)#dp     NB. starts of nested
ep=. (#e)+(0=ut)#dp           NB. starts of other
dp=. /:~ ~.0,sp,ep           NB. cut starts
ut=. }: 1 dp} (>:#y)#0        NB. cut mask
(dp i. sp);<ut <;.1 y        NB. nest indexes cut list
else.
  (i.0);<<y                  NB. empty arg result
end.
)

NB. delete trailing line feed if necessary: dlf 'ab',LF
dlf=: ] }.~ [: - (10{a.) = {:

extfreq=: 3 : 0

```

*NB.*extfreq v-- file extension frequency in descending order.*

NB.

NB. monad: ct =. rawfreq blclFiles

```
'ext cnt'=. ofreq s: tolower@('.'&afterlaststr)&.> y
(4 s: ext) ,. ' - ' , "1 ": ,.cnt
)
```

NB. boxes UTF8 names

```
fboxname=: ([: < 8 u: >) ::]
```

NB. 1 if file exists 0 otherwise

```
fexist=: 1:@(1!:4) ::0:@(fboxname&>)@boxopen
```

NB. 0's all but first 1 in runs of 1's - like (firstone) but differs for nulls

```
firstones=: > (0: , }:)
```

NB. get pure element text

```
geteletext=: ] betweenstrs~ [: tags [: alltrim [
```

NB. file name and extension from fully qualified file

```
justfileext=: ] #~ [: -. [: +./\ . '\'&=
```

NB. REFERENCE - standard z locale verb

```
list=: list_z_
```

NB. like (freq) but results in descending frequency
ofreq=: [: (([: < [: \: [: ; 1 {]) { &.>]) ~. ; #/.~

NB. reads a file as a list of bytes
read=: 1!:1&([`<@.(32&>@ (3!:0)))

NB. sha-256 hash from bytes: sha256 'hash me again'
sha256=: 3&(128!:6)

sidecars=: 3 : 0

*NB.*sidecars v-- image raws with corresponding sidecar xmp files.*

NB.

NB. monad: btcl =. sidecars clDirectory

NB.

NB. p0=. 'c:/pictures/2022/idaho/01_jan/iphoneraw'

NB. sidecars p0

NB.

NB. dyad: btcl =. blcl sidecars clDirectory

NB.

NB. p0=. '/Users/johnbaker/Pictures/2022/Idaho/07_jul/d7500'

NB. p1=. 'C:\pictures\2022\North Rim Monument Valley\06_jun\d7500'

NB. (;'nef dng') sidecars p1 NB. only real raws

NB. image types considered "raws"

RAWFILETYPES sidecars y

:


```
raw=. x allraws y
```

```
NB. darktable sidecar file names are created by
NB. appending '.xmp' to the source file name
(fexist xmp) # raw,.xmp=.raw ,&.> <'.xmp'
)
```

```
NB. xml BEGIN and END tags
tags=: '<'&,@,&'>' ; '</'&,@,&'>'
```

```
timestamp=: 3 : 0
```

```
NB.*timestamp v-- formats timestamp as dd mmm yyyy hr:mn:sc
NB.
NB. monad: cl =. timestamp zu | nlTime
NB.
NB. timestamp '' NB. empty now
NB. timestamp 2007 9 16 NB. fills missing
NB. timestamp 1953 7 2 12 33
```

```
if. 0 = #y do. w=. 6!:0'' else. w=. y end.
r=. }: $ w
t=. 2 1 0 3 4 5 {"1 [ _6 [\ , 6 {"1 <. w
d=. '+++::' 2 6 11 14 17 }"1 [ 2 4 5 3 3 3 ": t
mth=. _3[\ ' JanFebMarAprMayJunJulAugSepOctNovDec'
d=. ,((1 {"1 t) { mth) 3 4 5 }"1 d
d=. '0' (I. d=' ') } d
```

```
d=. ' ' (I. d='+') } d
(r,20) $ d
)
```

```
titbranddir=: 3 : 0
```

```
NB.*titbranddir v-- brand eligible xmp files in directory.
```

```
NB.
```

```
NB. NOTE: this verb reads entire directories filled with large
```

```
NB. >20MB image raw files to compute SHA256 hashes for each
```

```
NB. image. It may take a minute or so depending on the size and
```

```
NB. number of images in a directory.
```

```
NB.
```

```
NB. monad: blcl =. titbranddir clDirectory
```

```
NB.
```

```
NB. r0=. 'c:\pictures\2022\North Rim Monument Valley\06_jun\d5100'
```

```
NB. r1=. '/Users/johnbaker/Pictures/2022/Idaho/07_jul/d7500'
```

```
NB. titbranddir r0
```

```
NB. "raws" with sidecar xmp
```

```
if. #ds=. sidecars y do.
```

```
  NB. insert file name & hash in title element
```

```
xmps=. titbrandxmp&.> <"1 ds
```

```
  NB. write branded xmp files
```

```
xmps wrecho&.> 1 {"1 ds
```

```
else.
```

```
0$<' ' NB. no eligible xmps
end.
)

titbrandxmp=: 3 : 0

NB.*titbrandxmp v-- brand xmp sidecar file with file name and
NB. hash of associated image.
NB.
NB. monad: clXmp =. titbrandxmp blImageXmpFiles
NB.
NB. xmp=. 'c:/pictures/2022/Idaho/07_jul/d7500/_DSC8496.NEF.xmp'
NB. ps=. xmp ;~ (-#'.xmp') }. xmp
NB. titbrandxmp ps
NB.
NB. ds=. sidecars 'c:/pictures/2022/North Rim Monument Valley/06_jun/d7500'
NB. xmps=. titbrandxmp&.> <"1 ds

xmp=. read xmp [ 'raw xmp'=. y

NB. single Dublin Core publisher and creator
NB. elements must exist to safely brand
dcp=. '</dc:publisher>'; '</dc:creator>'
if. -.1 1 -: +/"1 dcp E.&> <xmp do. xmp return. end.

NB. file name and sha256 brand
tit=. dlf ('/[~(fhash)~]/',shabrand raw) changestr XMPTITLEFRAG-.CR
```

NB. replace or insert title element

```
'idx ctmp'=. (tags 'dc:title') cutnestidx xmp
if. #idx do. ;(<tit) idx} ctmp
else.
  (pt ,~ pt beforestr xmp),LF,tit,pt afterstr xmp [ pt=. ;0{dcp
end.
)
```

NB. appends trailing line feed character if necessary

```
tlf=: ] , ((10{a.})"_ = {:) }. (10{a.})"_
```

NB. converts character strings to CRLF delimiter

```
toCRLF=: 2&}.@:;@:((13{a.}&,&.>@<;.1@((10{a.}&,)@toJ)
```

NB. converts character strings to host delimiter

```
toHOST=: toCRLF
```

NB. converts character strings to J delimiter LF

```
toJ=: ((10{a.}) I.@(e.&(13{a.}))@]} ]>@:(#~ -.@((13 10{a.})&E.@,))
```

```
tolower=: 3 : 0
```

*NB.*tolower v-- convert to lower case.*

NB.

NB. monad: cl =. tolower cl

```
x=. I. 26 > n=. ((65+i.26){a.) i. t=. ,y
($y) $ ((x{n) { (97+i.26){a.) x}t
)
```

```
NB. appends trailing / iff last character is not \ or /
tslash2=: ([: - '\/' e.~ {:) }. '/' ,~ ]
```

```
NB. standardizes path delimiter to windows back \ slash
winpathsep=: '\&(('/' I.@:= ])) }
```

```
NB. writes a list of bytes to file
write=: 1!:2 ]`<@.(32&>@{3!:0))
```

```
NB.POST_brandxmp post processor.
```

```
smoutput IFACE=: (0 : 0)
NB. (brandxmp) interface word(s): 20230521j120831
NB. -----
NB. audbranddir  NB. audit xmp/raw image directories
NB. sidecars    NB. image raws with corresponding sidecar xmp files
NB. titbranddir NB. brand eligible xmp files in directory
NB. titbrandxmp NB. brand xmp sidecar file with file name and hash of associated image
)

cocurrent 'base'
coinset   'brandxmp'
```

Index

afterlaststr, 7
afterstr, 7
allraws, 7
alltrim, 8
audbranddir, 8

beforestr, 11
betweenstrs, 11
boxopen, 11

changestr, 11
CR, 6
ctl, 13
cutnestidx, 13

dlf, 14

extfreq, 14

fboxname, 15
fexist, 15

firstones, 15

geteletext, 15

IFACE, 21
IFACEWORDSbrandxmp, 6

justfileext, 15

LF, 6
list, 15

ofreq, 16

RAWFILETYPES, 6
read, 16
ROOTWORDSbrandxmp, 7

sha256, 16
shabrand, 6
sidecars, 16

tags, 17

timestamp, 17
titbranddir, 18
titbrandxmp, 19
tlf, 20
toCRLF, 20
toHOST, 20
toJ, 20
tolower, 20
tslash2, 21

VMDbrandxmp, 7

winpathsep, 21
wrecho, 6
write, 21

XMPAUDITFILE, 7
XMPBRDEL, 5
XMPTITLEFRAG, 6
XMPWID, 7