

# Package ‘ImageArray’

May 16, 2026

**Type** Package

**Title** A framework for on-disk and in-memory image arrays

**Version** 1.1.0

**Date** 2026-01-20

**Description** ImageArray provides a framework for on-disk and in-memory image arrays, specifically for pyramidal images stored in HDF5, Zarr and life sciences image file formats (OME Bio-Formats).

**License** MIT + file LICENSE

**Encoding** UTF-8

**RoxygenNote** 7.3.3

**biocViews** Software, Visualization

**Depends** R (>= 4.5.0), EBImage

**Imports** methods, grDevices, S4Arrays, S4Vectors, DelayedArray, rhdf5, HDF5Array, Rarr, ZarrArray, magick, tools

**Suggests** testthat (>= 3.0.0), knitr, ggplot2, shiny, RBioFormats, BiocFileCache, BiocStyle

**Config/testthat/edition** 3

**VignetteBuilder** knitr

**URL** <https://github.com/BIMSBbioinfo/ImageArray>

**BugReports** <https://github.com/BIMSBbioinfo/ImageArray/issues>

**git\_url** <https://git.bioconductor.org/packages/ImageArray>

**git\_branch** devel

**git\_last\_commit** 052d69d

**git\_last\_commit\_date** 2026-04-28

**Repository** Bioconductor 3.24

**Date/Publication** 2026-05-15

**Author** Artür Manukyan [aut, cre, fnd] (ORCID:  
<<https://orcid.org/0000-0002-0441-9517>>)

**Maintainer** Artür Manukyan <artur-man@hotmail.com>

## Contents

BFArray-methods . . . . .	2
createImageArray . . . . .	3
create_zarr . . . . .	4
create_zarr_group . . . . .	4
getImageInfo . . . . .	5
ImageArray-methods . . . . .	6
writeImageArray . . . . .	9
<b>Index</b>	<b>11</b>

---

BFArray-methods	<i>BFArray constructor method</i>
-----------------	-----------------------------------

---

## Description

A function for creating objects of BFArray class

## Usage

```
BFArray(image.file, series, resolution)
```

```
## S4 method for signature 'BFArraySeed'
dim(x)
```

```
## S4 method for signature 'BFArraySeed'
type(x)
```

## Arguments

<code>image.file</code>	the path to the image read by RBioFormats
<code>series</code>	the series IDs of the pyramidal image, typical an integer starting from 1
<code>resolution</code>	the resolution IDs of the pyramidal image, typical an integer starting from 1
<code>x</code>	A BFArray object

## Value

A BFArray object

## Functions

- `dim(BFArraySeed)`: dim function for BFArray objects
- `type(BFArraySeed)`: type function for BFArray objects

**Examples**

```
# get image
library(RBioFormats)
img.file <- system.file("extdata",
                        "xy_12bit__plant.ome.tiff",
                        package = "ImageArray")
bfa <- BFAArray(img.file, series = 1, resolution = 2)
dim(bfa)
type(bfa)
```

---

createImageArray	<i>createImageArray</i>
------------------	-------------------------

---

**Description**

creates an object of ImageArray class

**Usage**

```
createImageArray(
  image,
  n.levels = NULL,
  series = NULL,
  resolution = NULL,
  max.pixel.threshold = 700,
  engine = "EBImage",
  verbose = FALSE
)
```

**Arguments**

image	the image
n.levels	the number of levels of the pyramidal image, typical an integer starting from 1
series	the series IDs of the pyramidal image, typical an integer starting from 1.
resolution	the resolution IDs of the pyramidal image, typical an integer starting from 1.
max.pixel.threshold	the maximum width and height pixel dimension that the lowest level of the image pyramid should have, thus the image will be downscaled two folds until both width and height is below the threshold. Default is 700 pixels. If n.levels is provided, this parameter will be ignored.
engine	the package to use for each image layer: either EBImage or magick-image
verbose	verbose

**Value**

An ImageArray object

**Examples**

```
# get image
library(EBImage)
img.file <- system.file("images", "sample.png", package="EBImage")

# create ImageArray
imgarray <- createImageArray(img.file, n.levels = 3)
imgarray_raster <- as.raster(imgarray, max.pixel.size = 300)
plot(imgarray_raster)
```

---

create_zarr	<i>create_zarr</i>
-------------	--------------------

---

**Description**

Create zarr store

**Usage**

```
create_zarr(store, version = "v2")
```

**Arguments**

store	the location of zarr store
version	zarr version

**Value**

'NULL'

**Examples**

```
store <- tempfile(fileext = ".zarr")
create_zarr(store)
```

---

create_zarr_group	<i>create_zarr_group</i>
-------------------	--------------------------

---

**Description**

Create a zarr group

**Usage**

```
create_zarr_group(store, name, version = "v2")
```

**Arguments**

store	the location of (zarr) store
name	name of the group
version	zarr version

**Value**

‘NULL‘

**Examples**

```
store <- tempfile(fileext = ".zarr")
create_zarr(store)
create_zarr_group(store, "gp")
```

---

<code>getImageInfo</code>	<i>getImageInfo</i>
---------------------------	---------------------

---

**Description**

get information of an ImageArray object

**Usage**

```
getImageInfo(object)
```

**Arguments**

object	an ImageArray object
--------	----------------------

**Value**

a data frame of width and height info

**Examples**

```
# get image
library(EBImage)
img.file <- system.file("images", "sample.png", package="EBImage")

# create ImageArray
dir.create(td <- tempfile())
output_h5 <- tempfile(fileext = ".h5")
imgarray <- writeImageArray(img.file,
                            output = output_h5,
                            name = "image",
                            verbose = FALSE)

getImageInfo(imgarray)

# create ImageArray
imgarray <- createImageArray(img.file, n.levels = 3)
imgarray_raster <- as.raster(imgarray, max.pixel.size = 300)
getImageInfo(imgarray)
```

## Description

Methods for ImageArray objects

## Usage

```
## S4 method for signature 'ImageArray,ANY,ANY,ANY'
x[i, j, ..., drop = FALSE]

## S4 method for signature 'ImageArray,numeric'
x[[i]]

## S4 replacement method for signature 'ImageArray,numeric'
x[[i, j, ...]] <- value

## S4 method for signature 'ImageArray'
dim(x)

## S4 method for signature 'ImageArray'
type(x)

## S4 method for signature 'ImageArray'
length(x)

ImageArray(meta, levels)

## S4 method for signature 'ImageArray'
realize(x, level = NULL, max.pixel.size = NULL, min.pixel.size = NULL)

## S3 method for class 'ImageArray'
as.raster(x, level = NULL, max.pixel.size = NULL, min.pixel.size = NULL, ...)

## S4 method for signature 'ImageArray'
rotate(x, angle)

## S4 method for signature 'ImageArray'
aperm(a, perm)

## S4 method for signature 'ImageArray'
negate(object)

## S4 method for signature 'ImageArray'
modulate(object, brightness)

## S4 method for signature 'ImageArray'
flip(x)

## S4 method for signature 'ImageArray'
```

```
flop(x)

## S4 method for signature 'ImageArray'
crop(object, index)

## S4 method for signature 'ImageArray'
axes(object)

## S4 method for signature 'ImageArray'
meta(object)

## S4 method for signature 'ImageArray'
path(object)

## S4 replacement method for signature 'ImageArray'
path(object) <- value
```

### Arguments

x, a, object	An ImageArray object
i, j, value	Depends on the usage [[, [[<- Here i is the level of the image pyramid. You can use the length function to get the number of the layers in the pyramid. When used with crop, arguments i and j are associated with indices of image dimensions (e.g. width, height)
...	Arguments passed to other methods
drop	ignored
meta	the metadata of the ImageArray object.
levels	levels of the pyramid image, typically a vector of integers starting with 1
level	level
max.pixel.size	maximum pixel size
min.pixel.size	minimum pixel size
angle	value between 0 and 360 for degrees to rotate
perm	perm
brightness	the brightness of the new image in percentage, e.g. 120
index	a named or unnamed list of indices for cropping/subsetting the

### Value

dim of the first level of the ImageArray object  
 type of ImageArray object  
 length of ImageArray object  
 An ImageArray object

## Functions

- `x[i]`: subset and crop for ImageArray objects
- `x[[i]`: Layer access for ImageArray objects
- ``[[` (x = ImageArray, i = numeric) <- value`: Layer access for ImageArray objects
- `dim(ImageArray)`: dimensions of an ImageArray
- `type(ImageArray)`: dimensions of an ImageArray
- `length(ImageArray)`: length of an ImageArray
- `ImageArray()`: ImageArray constructor method  
A function for creating objects of ImageArray class
- `realize(ImageArray)`: realize the array
- `as.raster(ImageArray)`: create a raster object
- `rotate(ImageArray)`: rotate image array to 90, 180, 270 degrees
- `aperm(ImageArray)`: permute image
- `negate(ImageArray)`: negate image
- `modulate(ImageArray)`: modulate image
- `flip(ImageArray)`: vertical flipping image
- `flop(ImageArray)`: horizontal flipping image
- `crop(ImageArray)`: cropping image
- `axes(ImageArray)`: get axes metadata of the ImageArray object
- `meta(ImageArray)`: get metadata of the ImageArray object
- `path(ImageArray)`: path of an ImageArray object
- `path(ImageArray) <- value`: replace method for `path(ImageArray)`

## Examples

```
# get image
library(EBImage)
img.file <- system.file("images", "sample.png", package="EBImage")

# create ImageArray
imgarray <- createImageArray(img.file, n.levels = 3)

# access layers
imgarray[[1]]
imgarray[[2]]

# dimensions and length
dim(imgarray)
length(imgarray)

# manipulate images
imgarray <- crop(imgarray, ind = list(100:200, 100:200))
imgarray <- crop(imgarray, ind = list(x = 10:20, y = 10:20))
imgarray <- rotate(imgarray, angle = 90)
imgarray <- flip(imgarray)
imgarray <- flop(imgarray)

# create ImageArray on disk as HDF5 format
```

```

dir.create(td <- tempfile())
output_h5 <- tempfile(fileext = ".h5")
imgarray <- writeImageArray(img.file,
                            output = output_h5,
                            name = "image",
                            verbose = FALSE)

# as.raster
imgarray_raster <- as.raster(imgarray)

# realize
imgarray <- realize(imgarray)

```

---

writeImageArray	<i>writeImageArray</i>
-----------------	------------------------

---

## Description

Writing image arrays on disk

## Usage

```

writeImageArray(
  image,
  output = "my_image",
  name = "",
  format = NULL,
  replace = FALSE,
  n.levels = NULL,
  chunkdim = NULL,
  level = NULL,
  engine = "EBImage",
  verbose = FALSE,
  ...
)

```

## Arguments

image	an Image object (EBImage), a magick object or the path to an image file,
output	output file name
name	name of the group
format	on disk format, either "h5" for HDF5 format, "zarr" for zarr format, or "in-memory" for in-memory ImageArray object. If not provided, the format will be inferred from the file extension of the output path.
replace	Should the existing file be removed or not
n.levels	the number of levels if the image supposed to be pyramidal.
chunkdim	The dimensions of the chunks to use for writing the data to disk.
level	The compression level to use for writing the data to disk.
engine	the package to use for each image layer: either EBImage or magick-image
verbose	verbose
...	additional parameters passed to <a href="#">createImageArray</a> .

**Value**

An ImageArray object

**Examples**

```
# get image
library(EBImage)
img.file <- system.file("images", "sample.png", package="EBImage")

# create ImageArray
dir.create(td <- tempfile())
output_h5 <- tempfile(fileext = ".h5")
imgarray <- writeImageArray(img.file,
                             output = output_h5,
                             name = "image",
                             verbose = FALSE)
imgarray_raster <- as.raster(imgarray)
plot(imgarray_raster)
```

# Index

- [, ImageArray, ANY, ANY, ANY-method  
(ImageArray-methods), 6
- [[, ImageArray, numeric-method  
(ImageArray-methods), 6
- [[<- , ImageArray, numeric-method  
(ImageArray-methods), 6
  
- aperm, ImageArray-method  
(ImageArray-methods), 6
- as.raster (ImageArray-methods), 6
- as.raster, ImageArray-method  
(ImageArray-methods), 6
- as.raster.ImageArray  
(ImageArray-methods), 6
- axes (ImageArray-methods), 6
- axes, ImageArray-method  
(ImageArray-methods), 6
  
- BFArray (BFArray-methods), 2
- BFArray-methods, 2
  
- create\_zarr, 4
- create\_zarr\_group, 4
- createImageArray, 3, 9
- crop (ImageArray-methods), 6
- crop, ImageArray-method  
(ImageArray-methods), 6
  
- dim, BFArraySeed-method  
(BFArray-methods), 2
- dim, ImageArray-method  
(ImageArray-methods), 6
  
- flip (ImageArray-methods), 6
- flip, ImageArray-method  
(ImageArray-methods), 6
- flop (ImageArray-methods), 6
- flop, ImageArray-method  
(ImageArray-methods), 6
  
- getImageInfo, 5
  
- ImageArray (ImageArray-methods), 6
- ImageArray-methods, 6
  
- length, ImageArray-method  
(ImageArray-methods), 6
  
- meta (ImageArray-methods), 6
- meta, ImageArray-method  
(ImageArray-methods), 6
- modulate (ImageArray-methods), 6
- modulate, ImageArray-method  
(ImageArray-methods), 6
  
- negate (ImageArray-methods), 6
- negate, ImageArray-method  
(ImageArray-methods), 6
  
- path (ImageArray-methods), 6
- path, ImageArray-method  
(ImageArray-methods), 6
- path<- , ImageArray-method  
(ImageArray-methods), 6
  
- realize (ImageArray-methods), 6
- realize, ImageArray-method  
(ImageArray-methods), 6
- rotate (ImageArray-methods), 6
- rotate, ImageArray-method  
(ImageArray-methods), 6
  
- type, BFArraySeed-method  
(BFArray-methods), 2
- type, ImageArray-method  
(ImageArray-methods), 6
  
- writeImageArray, 9