Images

Image Class

Note for Linux Users: In the current release of Wakanda, the Image API is not supported on Linux platforms.

The Image class of Wakanda manages and works with *Image* type objects on the server. These objects are:

- values of image type attributes in your datastore classes (see the **Image Attribute** section) or
- images that are loaded directly from disk using the **loadImage( )** method.

These objects have methods and properties that you can use to work with them and to get information about their contents.

Wakanda includes native support of image type objects server-side. Images that you work with through the datastore classes are stored in their original format without interpretation. You can also retrieve and modify metadata for the images. You can access this metadata as properties (see **meta**) and can modify it using the **saveMeta( )** method.

Note: On the client, you can display image type objects or image attributes easily by associating a datasource with the Image widget.

Supported Image Formats

Here is the list of image types supported natively by Wakanda on Windows and Mac OS. You can either pass a Mime type or an extension in the *type* parameter.

Format	Mime Type(s)	Extension(s)
JPEG	image/jpeg, image/jpg, image/pjpeg, image/jpe_	.jpg, .jif, .jpeg, .jpe
PNG	image/png, image/x-png	.png
BMP	image/bmp, image/x-bmp	.bmp, .dib, .rle
GIF	image/gif	.gif
TIFF	image/tiff, image/x-tiff	.tif, .tiff
Windows Metafile (Windows only)	image/x-emf	.emf
PDF (Mac OS only)	application/pdf, application/x-pdf	.pdf
SVG	image/svg+xml	.svg

Note: This list contains formats that are managed by default regardless of the operating system. Additional formats may be available depending on the Wakanda server platform and on the elements installed.

height

Description

The **height** property returns the height (in pixels) of the *Image* object.

length

Description

The **length** property returns the size (expressed in bytes) of the *Image* type object.

Note: You can use either the size or ***length*** *property to return the size of the Image.*

size

Description

The **size** property returns the size (expressed in bytes) of the *Image* type object.

Note: You can use either the size or ***length*** *property to return the size of the Image.*

width

Description

The **width** property returns the width (in pixels) of the *Image* object.

meta

Description

The **meta** property returns an object made up of one or more sub-objects containing the metadata associated with the *Image* object. If the image does not contain any metadata, the property returns an empty object.

Metadata is additional information that is inserted in images. Wakanda works with four standard metadata types: EXIF, GPS, IPTC, and TIFF. If metadata is found in the image, the **meta** object contains as many sub-objects as there are different types of metadata, named EXIF, GPS, IPTC, and TIFF.

Note: For a detailed description of these metadata types, refer to the following documents: http://www.iptc.org/std/IIM/4.1/specification/IIMV4.1.pdf (IPTC) and http://exif.org/Exif2-2.PDF (TIFF, EXIF and GPS).

Each sub-object consists of a set of properties/values described in the paragraph below.

EXIF

Exchangeable image file format (EXIF) is the image file format specification used by digital cameras. This specification uses the existing JPEG, TIFF Rev. 6.0, and RIFF WAV file formats with the addition of specific metadata tags. It is not supported by JPEG 2000, PNG, or GIF.

Here is a list of properties and possible values for the EXIF metadata:

Properties	Possible values
ApertureValue	Number (APEX value)
BrightnessValue	Number (APEX value)
ColorSpace	2 (Adobe RGB), 1 (s RGB), -1 (Uncalibrated)
ComponentsConfiguration	6 (B), 2 (Cb), 3 (Cr), 5 (G), 4 (R), 0 (Unused), 1 (Y)

CompressedBitsPerPixel	Number
Contrast	2 (High), 1 (Low), 0 (Normal)
CustomRendered	0 (Normal), 1 (Custom)
DateTimeDigitized	XML Datetime
DateTimeOriginal	XML Datetime
DigitalZoomRatio	Number
ExifVersion	String (4 digits)
ExposureBiasValue	Number
ExposureIndex	Number
ExposureModus	0 (Auto), 2 (Auto Bracket), 1 (Manual)
ExposureProgram	1 (Manual), 5 (Action), 3 (Aperture Priority AE), 5 (Creative), 8 (Landscape), 7 (Exposure Portrait), 2 (Program AE), 4 (Shutter Speed Priority AE)
ExposureTime	Number
FNumber	Number
FileSource	3 (Digital Camera), 1 (Film Scanner), 2 (Reflection Print Scanner)
Flash	3 (Auto Mode), 1 (Compulsory Flash Firing), 2 (Compulsory Flash Suppression), 0 (Unknown), 3 (Detected), 0 (No Detection Function), 2 (Not Detected), 1 (Reserved)
FlashEnergy	Number
Flash/Fired	Boolean
Flash/FunctionPresent	Boolean
Flash/Mode	3 (Auto Mode), 1 (Compulsory Flash Firing), 2 (Compulsory Flash Suppression), 0 (Unknown)
FlashPixVersion	String (4 values)
Flash/RedEyeReduction	Boolean
Flash/ReturnLight	3 (Detected), 0 (No Detection Function), 2 (Not Detected), 1 (Reserved)
FocalLenIn35mmFilm	Number
FocalLength	Number
FocalPlaneResolutionUnit	Number
FocalPlaneXResolution	Number
FocalPlaneYResolution	Number
GainControl	4 (High Gain Down), 2 (High Gain Up), 3 (Low Gain Down), 1 (Low Gain Up), 0 (None)
Gamma	Number
ImageUniqueID	Text
ISOSpeedRatings	Number
LightSource	0 (Unknown), 10 (Cloudy), 14 (Cool White Fluorescent), 23 (D50), 20 (D55), 21 (D65), 22 (D75), 1 (Daylight), 12 (Daylight Fluorescent), 13 (Day White Fluorescent), 9 (Fine Weather), 4 (Flash), 2 (Light Fluorescent), 24 (ISOStudio Tungsten), 255 (Other), 11 (Shade), 17 (Standard Light A), 18 (Standard Light B), 19 (Standard Light C), 3 (Tungsten), 15 (White Fluorescent)
MakerNote	Text
MaxApertureValue	Number
MeteringMode	255 (Other), 1 (Average), 2 (Center Weighted Average), 5 (Multi Segment), 4 (Multi Spot), 6 (Partial), 3 (Spot)
PixelXDimension	Number
PixelYDimension	Number
RelatedSoundFile	Text
Saturation	EXIF High, EXIF Low, EXIF Normal
SceneCaptureType	EXIF Scene Landscape, EXIF Night, EXIF Scene Portrait, EXIF Standard
SceneType	Longint
SensingMethod	EXIF Color Sequential Area, EXIF Color Sequential Linear, EXIF Not Defined, EXIF One Chip Color Area, EXIF Three Chip Color Area, EXIF Trilinear, EXIF Two Chip Color Area
Sharpness	2 (High), 1 (Low), 0 (Normal)
ShutterSpeedValue	Number
SpectralSensitivity	Text
SubjectArea	String (2, 3 or 4 values)
SubjectDistRange	0 (Unknown), 2 (Close), 3 (Distant), 1 (Macro)
SubjectDistance	Number
SubjectLocation	String (2 values)
UserComment	String
WhiteBalance	0 (Auto), 1 (Manual)

GPS

Here is a list of properties and possible values for the GPS (geolocation) metadata:

Properties	Possible values
Altitude	0 (Above Sea Level), 1 (Below Sea Level)
AltitudeRef	0 (Above Sea Level), 1 (Below Sea Level)
AreaInformation	Text
DateTime	XML Datetime
DestBearing	Text (1 character)
DestBearingRef	Text (1 character)
DestDistance	Text (1 character)
DestDistanceRef	Text (1 character)
DestLatitude	Text
DestLatitude/Deg	Real
DestLatitude/Dir	Text (1 character)

DestLatitude/Min	Number
DestLatitude/Sec	Number
DestLongitude	Text
DestLongitude/Deg	Number
DestLongitude/Dir	Text (1 character)
DestLongitude/Min	Number
DestLongitude/Sec	Number
Differential	1 (Correction Applied), 0 (Correction Not Applied)
DOP	Number
ImgDirection	"M" (Magnetic north), "T" (True north)
ImgDirectionRef	"M" (Magnetic north), "T" (True north)
Latitude	"N" (North), "S" (South)
Latitude/Deg	Number
Latitude/Dir	"N" (North), "S" (South)
Latitude/Min	Number
Latitude/Sec	Number
Longitude	"W" (West), "E" (East)
Longitude/Deg	Number
Longitude/Dir	"W" (West), "E" (East)
Longitude/Min	Number
Longitude/Sec	Number
MapDate	Texte
MeasureMode	2 (2D), 3 (3D)
ProcessingMethod	Text
Satellites	Text
Speed	"K" (km h), "M" (miles h), "K" (knots h)
SpeedRef	"K" (km h), "M" (miles h), "K" (knots h)
Status	"A" (Measurement in progress), "V" (Measurement Interoperability)
Track	Number (0.00..359.99)
TrackRef	String (1 character)
VersionID	String (4 characters)

IPTC

IPTC metadata attributes are widely used and supported by many image creation and manipulation programs. Almost all the IPTC metadata attributes are supported by the Exchangeable image file format (EXIF), the image file format specification used by digital cameras. IPTC metadata can be embedded into JPEG/Exif or TIFF formatted image files. Other file formats such as JPEG2000, Portable Network Graphics, and GIF do not support IPTC metadata.

Here is a list of properties and possible values for the IPTC metadata:

Properties	Possible values
IBylin	Text
BylineTitle	Text
CaptionAbstract	Text
Category	Text
City	Text
Contact	Text
ContentLocationCode	Text
ContentLocationName	Text
CopyrightNotice	Text
CountryPrimaryLocationCode	Text
CountryPrimaryLocationName	Text
Credit	Text
DateTimeCreated	XML Datetime
DigitalCreationDateTime	XML Datetime
EditStatus	Text
ExpirationDateTime	XML Datetime
FixtureIdentifier	Text
Headline	Text
ImageOrientation	Text
ImageType	Text
Keywords	Text
LanguageIdentifier	Text
ObjectAttributeReference	Text
ObjectCycle	Text
ObjectName	Text
OriginalTransmissionReference	Text
OriginatingProgram	Text
ProgramVersion	Text
ProvinceState	Text
ReleaseDateTime	XML Datetime
Scene	11900 (Action), 11200 (Aerial View), 11800 (Close Up), 10700 (Couple), 11900 (Exterior View), 10300 (Full Length), 11000 (General View), 10900 (Group), 10200 (Half Length), 10100 (Headshot), 11700 (Interior View), 12400 (Movie Scene), 11400 (Night Scene), 12300 (Off Beat), 111000 (Panoramic View), 12000 (Performing), 12100 (Posing), 10400 (Profile), 10500 (Rear

	View), 11500 (Satellite), 10600 (Single), 12200 (Symbolic), 10800 (Two), 11300 (Under Water)
Source	Text
SpecialInstructions	Text
StarRating	Number
SubLocation	Text
SubjectReference	Number
SupplementalCategory	String
Urgency	Number
WriterEditor	String

TIFF

Tagged Image File Format (TIFF) is a file format for storing images, popular among those using Apple Macintosh, such as graphic artists, the publishing industry as well as amateur and professional photographers. Developers can request a block of "private tags" to include their own proprietary information inside a TIFF file without causing problems for file interchange. TIFF readers are required to ignore tags they do not recognize, and a developer's private tags are guaranteed not to clash with anyone else's tags or with the standard set of tags defined in the specification.

Here is a list of properties and possible values for TIFF metadata:

Properties	Possible values
Artist	Text
Compression	8 (Adobe Deflate), 32771 (CCIRLEW), 2 (CCITT1D), 32947 (DCS), 32946 (Deflate), 32769 (Epson ERF), 32898 (IT8BL), 32895 (IT8CTPAD), 32896 (IT8LW), 32897 (IT8MP), 34661 (JBIG), 9 (JBIGB&W), 10 (JBIGColor), 7 (JPEG), 34712 (JPEG2000), 6 (JPEGThumbs Only), 262 (Kodak262), 65000 (Kodak DCR), 32867 (Kodak KDC), 5 (LZW), 34718 (MDIBinary Level Codec), 34719 (MDIProgressive Transform Codec), 34720 (MDIVector), 32766 (Next), 34713 (Nikon NEF), 32773 (Pack Bits), 65535 (Pentax PEF), 32908 (Pixar Film), 32909 (Pixar Log), 34676 (SGILog), 34677 (SGILog24), 32767 (Sony ARW), 3 (T4Group3Fax), 4 (T6Group4Fax), 32809 (Thunderscan), 1 (Uncompressed)
Copyright	Text
DateTime	XML Datetime
DocumentName	Text
HostComputer	Text
ImageDescription	Text
Make	Text
Model	Text
Orientation	1 (Horizontal), 2 (Mirror Horizontal), 5 (Mirror Horizontal And Rotate270CW), 7 (Mirror Horizontal And Rotate90CW), 4 (Mirror Vertical), 3 (Rotate180), 8 (Rotate270CW), 6 (Rotate90CW)
PhotometricInterpretation	1 (Black Is Zero), 8 (CIELab), 5 (CMYK), 32803 (Color Filter Array), 9 (ICCLab), 10 (ITULab), 34892 (Linear Raw), 32844 (Pixar Log L), 32845 (Pixar Log Luv), 2 (RGB), 3 (RGBPalette), 4 (Transparency Mask), 0 (White Is Zero), 6 (YCb Cr)
ResolutionUnit	3 (CM), 2 (Inches), 4 (MM), 1 (None), 5 (UM)
Software	Text
XResolution	Number
YResolution	Number

Example

Below is an example of loading an image including TIFF and EXIF metadata:

```
var img = loadImage("c:/temp/Tulips.jpg");
var imgMeta = img.meta;
    // imgMeta contains (for example):
    // { TIFF: { PhotometricInterpretation: "2", Orientation: "1", XResolution: "96/1", 4 more},
    // EXIF: { ExifVersion: "0221", DateTimeOriginal: "0000-00-00T00:00:00Z", 5 more} }
```

save( )

void **save**(String | File *file* [, String *type*])

Parameter	Type	Description
file	String, File	Path for the file to create or File object
type	String	Format of image to save

Description

The **save( )** method stores the *Image* object in a file.

You can store the image directly in a file on disk or in a *File* object:

- If you pass an absolute path to *file*, the image is stored in a file on disk at the location specified. You can also pass the extension of the file to create.
- If you pass a reference to a *File* object in *file*, the image is stored in a *File* object that you can then (if desired) store on disk (for more information about *File* objects, refer to the documentation for **Files and Folders**).

In the *type* parameter, pass a string indicating the format of the image to save. You can pass either a Mime type (e.g., "image/jpg"), or an extension (e.g., ".jpg"). In most cases, it is recommended that you pass a Mime type. The list of image formats supported by Wakanda Server is located in the **Image Class** section.

Note: By default, if you pass a path and omit the type parameter, Wakanda tries to determine the image format based on the extension of the file parameter.

saveMeta( )

void **saveMeta**(Object *meta*)

Parameter	Type	Description

meta	Object	Object containing the metadata to be modified

Description

The **saveMeta( )** method modifies metadata found in the *Image* object. Metadata is additional information that is inserted in an image. Wakanda works with four types of standard metadata: EXIF, GPS, IPTC, and TIFF. You can find out the current metadata for an image by using the **meta** property.

In *meta*, pass an object containing at least one of the EXIF, GPS, IPTC, or TIFF members including property/value pairs. Only the properties contained in the object are updated, any other metadata is not changed. For a complete list of properties and values that can be modified for each type of metadata, refer to the description of the **meta** property.

Note: This method cannot be used with Image objects saved in datastore attributes. For more information on how to edit and save metadata in image attributes, please refer to the Editing an Image Attribute Metadata paragraph.

Example

We want to add the keywords "vacation" and "snow" in the IPTC metadata for an image stored on disk:

```
var img = loadImage("c:/test/img00210.jpg");  // load the image
var newMeta = { IPTC:{Keywords: ["vacation", "snow"]}} ;  // create the metadata to add
img.saveMeta ( newMeta );  // update metadata
img.save ("c:/test/img00210.jpg") ;  // save the information in the file
```

setPath( )

void **setPath**(File | String *file*)

Parameter	Type	Description
file	File, String	Image file object or path

Description

The **setPath( )** method allows you to associate a *file* path to an *Image* object.

This feature is designed for image attributes. When you associate a *file* to an *Image* and then assign the *Image* to an image attribute, only the referenced file is stored in the attribute, just like a direct assignment by reference. When you save the entity, the referenced file is also generated from the *Image* and saved on disk (if a file was already existing at the location, it is replaced). The referenced *file* is used each time you need to access the image attribute.

The **setPath( )** method is useful for example when you parse a picture folder and want to reference each picture it contains without having to process them all.

For more information about the different ways to assign an image to an attribute, please refer to the see **Assigning an Image to an Attribute** section.

Example

We want to create a thumbnail from a loaded image, and store it as a reference in an attribute:

```
var pictFile = File ("C:/Wakanda/facebook/johndoe.jpg"); // get a file reference to a pict file on disk
var myPict = loadImage (pictFile);  // load the image
var thumb = myPict.thumbnail(300,200,4); // create a thumbnail from the image
var thumbFile = File(pictFile.getParent(), pictFile.nameNoExt+"_thumb."+pictFile.extension); // reference a new file
thumb.setPath(thumbFile); // set the file path to the thumbnail image
var p = new ds.Person( //create a new Person entity
    {
        name: "Doe",
        firstName: "John",
        photo: thumb //assign the image reference to the attribute
    });
p.save(); // save the person
// the thumbnail file is also automatically saved on disk
```

thumbnail( )

Image **thumbnail**([Number *width* [, Number *height*[, Number *mode*]]])

Parameter	Type	Description
width	Number	Thumbnail width in pixels, default value = 48
height	Number	Thumbnail height in pixels, default value = 48
mode	Number	Thumbnail creation mode, default value = 6 (scaled to fit proportional and centered)
Returns	Image	Resulting thumbnail

Description

The thumbnail( ) method returns a thumbnail of the source image. Thumbnails are useful when previewing images.

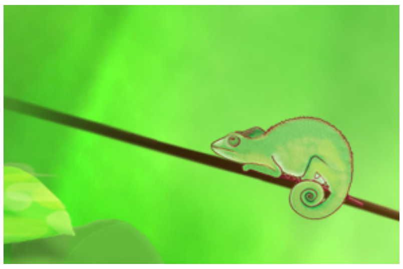
The optional *width* and *height* parameters allow you to define the required thumbnail size in pixels. If you omit these parameters, the default thumbnail size is 48 by 48 pixels.

The optional *mode* parameter sets the thumbnail creation mode, i.e., the reduction mode. Three modes are available:

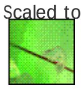
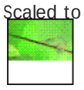
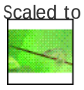
2	Scaled to fit (the proportions are not maintained)
5	Scaled to fit proportionally (proportions are preserved, the image is aligned to the top left)
6	Scaled to fit proportionally and centered

If you do not enter any parameter, the “Scaled to fit proportionaly and centered” mode (6) is applied by default. Below is an illustration of the various modes:

Source picture



Resulting thumbnails (48x48)

- Scaled to fit = 2
  
- Scaled to fit proportionally = 5
  
- Scaled to fit proportionally centered = 6 (default mode)
  

Example

Creating a 50 x 50 pixel thumbnail based on an image stored in the application folder:

```
var newThumb, basePath;
basePath = application.getFolder().path; // building image path
newThumb = loadImage(basePath + "img1.jpg"); // loading image
newThumb = newThumb.thumbnail(50, 50); // creating thumbnail
newThumb.save(basePath + "img1_thumb.jpg"); // saving image
```

Image Constructor

loadImage( )

Image **loadImage**(File | String *file*)

Parameter	Type	Description
file	File, String	Image file object or path
Returns	Image	Image object

Description

The **loadImage( )** method loads the image stored in a file referenced by the *file* parameter and returns an *image* object. You can pass either a *File* object or a string containing a standard file path in the *file* parameter (use the "/" as folder separator).

Note: In the current version of Wakanda, you have to pass an absolute path in the file parameter.

If the file does not contain a valid image or if the file reference is invalid, the method returns *null*.

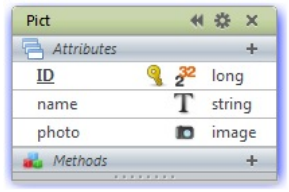
For more information about Wakanda image object manipulation, refer to the **Images** class description.

Note for Linux Users: In the current version of Wakanda, the Image API is not supported on Linux platforms.

Example

This example loads the image in a JPG file stored on the server and stores it in a new entity in the Pict class (in the photo attribute).

Here is the (simplified) datastore class:



ID	long
name	string
photo	image

```
var mypict = loadImage ("C:/Wakanda/Solutions/mysolution/Tulips.jpg"); // load the image from file
var p = new Pict(); // create a new entity in the Pict datastore class
p.name = "Flower"; // name the image
p.photo = mypict; // put the image in the photo attribute
p.save(); // save the entity
```