Image Field Type

Field Type identifier: `ezimage`
Validators: File size
Value object: `\eZ\Publish\Core\FieldType\Image\Value`
Associated Services: `ezpublish.fieldType.ezimage.variation_service`

The Image Field Type allows you to store an image file.

A **variation service** handles conversion of the original image into different formats and sizes through a set of preconfigured named variations: large, small, medium, black & white thumbnail, etc.

PHP API Field Type

Value object

The `value` property of an Image Field will return an `\eZ\Publish\Core\FieldType\Image\Value` object with the following properties:

Properties

Property	Type	Example	Description
`id`	string	`0/8/4/1/1480-1-eng-GB/image.png`	The image's unique identifier. Usually the path, or a part of the path. To get the full path, use `URI` property.
`alternativeText`	string	`This is a piece of text`	The alternative text, as entered in the Field's properties
`fileName`	string	`image.png`	The original image's filename, without the path
`fileSize`	int	`37931`	The original image's size, in bytes
`uri`	string	`var/ezdemo_site/storage/images/0/8/4/1/1480-1-eng-GB/image.png`	The original image's URI
`imageId`	string	`240-1480`	A special image ID, used by REST

Image Variations

Using the variation Service, variations of the original image can be obtained. Those are `\eZ\Publish\SPI\Variation\Values\ImageVariation` objects with the following properties:

Property	Type	Example	Description
`width`	int	`640`	The variation's width in pixels
`height`	int	`480`	The variation's height in pixels
`name`	string	`medium`	The variation's identifier
`info`	mixed		Extra info, such as EXIF data
`fileSize`	int		

`mimeType`	string		
`fileName`	string		
`dirPath`	string		
`uri`	string		The variation's uri
`lastModified`	DateTime		When the variation was last modified

Field Defintion options

The Image Field Type supports one FieldDefinition option: the maximum size for the file.

Using an Image Field

Template Rendering

When displayed using `ez_render_field`, an Image Field will output this type of HTML:

```
<img
src="var/ezdemo_site/storage/images/0/8/4/1/1480-1-eng-G
B/image_medium.png" width="844" height="430"
alt="Alternative text" />
```

The template called by the **ez_render_field()** Twig function while rendering a Image field accepts the following the parameters:

Parameter	Type	Default	Description
`alias`	`string`	`"original"`	The image variation name, must be defined in your site access `image_variations` settings. Defaults to "original", the originally uploaded image.
`width`	`int`		Optionally to specify a different width set on the image html tag then then one from image alias.
`height`	int		Optionally to specify a different height set on the image html tag then then one from image alias.
`class`	`string`		Optionally to specify a specific html class for use in custom javascript and/or css.

Example:

```
{{ ez_render_field( content, 'image', { 'parameters':{
'alias': 'imagelarge', 'width': 400, 'height': 400 } } )
}}
```

The raw Field can also be used if needed. Image variations for the Field's content can be obtained using the `ez_image_alias` Twig helper:

```
{% set imageAlias = ez_image_alias( field, versionInfo,
'medium' ) %}
```

The variation's properties can be used to generate the required output:

```
<img src="{{ asset( imageAlias.uri ) }}" width="{{
imageAlias.width }}" height="{{ imageAlias.height }}"
alt="{{ field.value.alternativeText }}" />
```

With the REST API

Image Fields within REST are exposed by the `application/vnd.ez.api.Content` media-type . An Image Field will look like this:

inputUri
From 5.2 version, new images must be input using the `inputUri` property from `Image\Value`.

The keys `id` and `path` still work, but a deprecation warning will be thrown.

Version >= 5.2

```
<field>
    <id>1480</id>

<fieldDefinitionIdentifier>image</fieldDefinitionIdentif
ier>
    <languageCode>eng-GB</languageCode>
    <fieldValue>
        <value
key="inputUri">/var/ezdemo_site/storage/images/0/8/4/1/1
480-1-eng-GB/kidding.png</value>
        <value key="alternativeText"></value>
        <value key="fileName">kidding.png</value>
        <value key="fileSize">37931</value>
        <value key="imageId">240-1480</value>
        <value
key="uri">/var/ezdemo_site/storage/images/0/8/4/1/1480-1
-eng-GB/kidding.png</value>
        <value key="variations">
            <value key="articleimage">
                <value
key="href">/api/ezp/v2/content/binary/images/240-1480/va
riations/articleimage</value>
            </value>
            <value key="articlethumbnail">
                <value
key="href">/api/ezp/v2/content/binary/images/240-1480/va
riations/articlethumbnail</value>
            </value>
        </value>
    </fieldValue>
</field>
```

Before 5.2 Expand source

```
<field>
    <id>1480</id>

<fieldDefinitionIdentifier>image</fieldDefinitionIdentif
ier>
    <languageCode>eng-GB</languageCode>
    <fieldValue>
        <value
key="id">var/ezdemo_site/storage/images/0/8/4/1/1480-1-e
ng-GB/kidding.png</value>
        <value
key="path">/var/ezdemo_site/storage/images/0/8/4/1/1480-
1-eng-GB/kidding.png</value>
        <value key="alternativeText"></value>
        <value key="fileName">kidding.png</value>
        <value key="fileSize">37931</value>
        <value key="imageId">240-1480</value>
        <value
key="uri">/var/ezdemo_site/storage/images/0/8/4/1/1480-1
-eng-GB/kidding.png</value>
        <value key="variations">
            <value key="articleimage">
                <value
key="href">/api/ezp/v2/content/binary/images/240-1480/va
riations/articleimage</value>
            </value>
            <value key="articlethumbnail">
                <value
key="href">/api/ezp/v2/content/binary/images/240-1480/va
riations/articlethumbnail</value>
            </value>
        </value>
    </fieldValue>
</field>
```

Children of the fieldValue node will list the general properties of the Field's original image (fileSize, fileName, inputUri, etc.), as well as variations. For each variation, a uri is provided. Requested through REST, this resource will generate the variation if it doesn't exist yet, and list the variation details:

```
<ContentImageVariation
media-type="application/vnd.ez.api.ContentImageVariation
+xml"
href="/api/ezp/v2/content/binary/images/240-1480/variati
ons/tiny">

<uri>/var/ezdemo_site/storage/images/0/8/4/1/1480-1-eng-
GB/kidding_tiny.png</uri>
  <contentType>image/png</contentType>
  <width>30</width>
  <height>30</height>
  <fileSize>1361</fileSize>
</ContentImageVariation>
```

From PHP code

Getting an image variation

The variation service, `ezpublish.fieldType.ezimage.variation_service`, can be used to generate/get variations for a Field. It expects a VersionInfo, the Image Field and the variation name, as a string (`large`, `medium`, etc.)

```
$variation = $imageVariationHandler->getVariation(
    $imageField, $versionInfo, 'large'
);

echo $variation->uri;
```

Manipulating image content

From PHP

As for any Field Type, there are several ways to input content to a Field. For an Image, the quickest is to call `setField()` on the ContentStruct:

```
$createStruct = $contentService->newContentCreateStruct(
    $contentTypeService->loadContentType( 'image' ),
    'eng-GB'
);

$createStruct->setField( 'image', '/tmp/image.png' );
```

In order to customize the Image's alternative texts, you must first get an Image\Value object, and set this property. For that, you can use the `Image\Value::fromString()` method that accepts the path to a local file:

```
$createStruct = $contentService->newContentCreateStruct(
    $contentTypeService->loadContentType( 'image' ),
    'eng-GB'
);

$imageField =
\eZ\Publish\Core\FieldType\Image\Value::fromString(
'/tmp/image.png' );
$imageField->alternativeText = 'My alternative text';
$createStruct->setField( 'image', $imageField );
```

You can also provide a hash of `Image\Value` properties, either to `setField()`, or to the constructor:

```
$imageValue = new
\eZ\Publish\Core\FieldType\Image\Value(
    array(
        'id' => '/tmp/image.png',
        'fileSize' => 37931,
        'fileName' => 'image.png',
        'alternativeText' => 'My alternative text'
    )
);

$createStruct->setField( 'image', $imageValue );
```

From REST

The REST API expects Field values to be provided in a hash-like structure. Those keys are identical to those expected by the `Image\Value` constructor: `fileName`, `alternativeText`. In addition, image data can be provided using the `data` property, with the image's content encoded as base64.

Creating an image Field

```
<?xml version="1.0" encoding="UTF-8"?>
<ContentCreate>
    <!-- [...metadata...] -->

    <fields>
        <field>
            <id>247</id>

<fieldDefinitionIdentifier>image</fieldDefinitionIdentif
ier>
            <languageCode>eng-GB</languageCode>
            <fieldValue>
                <value
key="fileName">rest-rocks.jpg</value>
                <value
key="alternativeText">HTTP</value>
                <value
key="data"><![CDATA[/9j/4AAQSkZJRgABAQEAZABkAAD/2wBDAAIB
AQIBAQICAgICAgICAwUDAwMDAwYEBAMFBwYHBwcG
BwcICQsJCAgKCAcHCg0KCgsMDAwMBwkODw0MDgsMDAz/2[...]</valu
e>
            </fieldValue>
        </field>
    </fields>
</ContentCreate>
```

Updating an image Field

Updating an Image Field requires that you re-send existing data. This can be done by re-using the Field obtained via REST, **removing the variations key**, and updating `alternativeText`, `fileN ame` or `data`. If you do not want to change the image itself, do not provide the `data` key.

```
<?xml version="1.0" encoding="UTF-8"?>
<VersionUpdate>
    <fields>
        <field>
            <id>247</id>

<fieldDefinitionIdentifier>image</fieldDefinitionIdentif
ier>
            <languageCode>eng-GB</languageCode>
            <fieldValue>
                <value
key="id">media/images/507-1-eng-GB/Existing-image.png</v
alue>
                <value key="alternativeText">Updated
alternative text</value>
                <value
key="fileName">Updated-filename.png</value>
   </fieldValue>
        </field>
    </fields>
</VersionUpdate>
```

Naming

Each storage engine determines how image files are named.

Legacy Storage Engine naming

Images are stored within the following directory structure:

`<varDir>/<StorageDir>/<ImagesStorageDir>/<FieIdId[-1]>/<FieIdId[-2]>/<FieIdId[-3]>/<FieIdId[-4]>/<FieldId>-<VersionNumber>-<LanguageCode>/`

With the following values:

- `VarDir` = var (default)
- `StorageDir` = `storage` (default)
- `ImagesStorageDir` = `images` (default)
- `FieldId` = `1480`
- `VersionNumber` = `1`
- `LanguageCode` = `eng-GB`

Images will be stored to `web/var/ezdemo_site/storage/images/0/8/4/1/1480-1-eng-GB`.

> Using the field ID digits in reverse order as the folder structure maximizes sharding of files through multiple folders on the filesystem.

Within this folder, images will be named like the uploaded file, suffixed with an underscore and the variation name:

- MyImage.png
- MyImage_large.png
- MyImage_rss.png