# The Image FieldType

> (i) FieldType identifier: `ezimage`
> Validators: File size
> Value object: `\eZ\Publish\Core\FieldType\Image\Value`
> Associated services: `ezpublish.fieldType.ezimage.variation_service`

The image FieldType 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 pre-configured named variations: large, small, medium, black & white thumbnail...

## Value object

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

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 |

## FieldType options

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

## Using an Image Field

### In templates

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-GB/image_medium.png"
width="844" height="430" alt="Alternative text" />
```

The size and height are picked from the used variation's attributes. By default, it will use the `original` variation, but any existing variation can be set using the `variation` parameter:

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

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 inputed 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</fieldDefinitionIdentifier>
    <languageCode>eng-GB</languageCode>
    <fieldValue>
        <value
key="inputUri">/var/ezdemo_site/storage/images/0/8/4/1/1480-1-eng-GB/kidding.png</valu
e>
        <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/variations/articleimage</value>
            </value>
            <value key="articlethumbnail">
                <value
key="href">/api/ezp/v2/content/binary/images/240-1480/variations/articlethumbnail</val
ue>
            </value>
        </value>
    </fieldValue>
</field>
```

```
<field>
    <id>1480</id>
    <fieldDefinitionIdentifier>image</fieldDefinitionIdentifier>
    <languageCode>eng-GB</languageCode>
    <fieldValue>
        <value
key="id">var/ezdemo_site/storage/images/0/8/4/1/1480-1-eng-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/variations/articleimage</value>
            </value>
            <value key="articlethumbnail">
                <value
key="href">/api/ezp/v2/content/binary/images/240-1480/variations/articlethumbnail</val
ue>
            </value>
        </value>
    </fieldValue>
</field>
```

Children of fieldValue will list the general properties of the Field's original image (fileSize, fileName, inputUri...), as well as variations. For each variation, an 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/variations/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`...)

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

echo $variation->uri;
```

# Manipulating image content

## From PHP

As for any fieldtype, 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 an 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`, `fileSize`, `alternativeText`. In addition, a `data` key is also expected, with the image's content as a base 64 encoded string.

This is valid for both updating and creating:

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

    <fields>
        <field>
            <fieldDefinitionIdentifier>image</fieldDefinitionIdentifier>
            <languageCode>eng-GB</languageCode>
            <fieldValue>
                <value key="fileName">rest-rocks.jpg</value>
                <value key="fileSize">17589</value>
                <value key="alternativeText">HTTP</value>
                <value
key="data"><![CDATA[/9j/4AAQSkZJRgABAQEAZABkAAD/2wBDAAIBAQIBAQICAgICAgICAwUDAwMDAwYEBA
MFBwYHBwcG
BwcICQsJCAgKAcHCg0KCgsMDAwMBwkODw0MDgsMDAz/2[...]]></value>
            </fieldValue>
        </field>
    </fields>
</ContentCreate>
```

# 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

# Changelog

| Version | Description |
| --- | --- |
| 5.2 | adding inputURI for input by API |

| 5.4<br>2014.09 | id property contains only the path of the image, not the full path |