

The BinaryFile FieldType

This FieldType represents and handles a binary file. It also counts the number of times the file has been downloaded from the `content/download` module.

| Name | Internal name | Expected input | Output |
|------------|---------------|----------------|--------|
| BinaryFile | ezbinaryfile | Mixed | Mixed |

Table of contents:

- [Description](#)
- [BinaryFile Value Object API](#)
- [BinaryFile hash format](#)
- [REST API specifics](#)
 - [Reading content: url property](#)
 - [Creating content: data property](#)

Description

This FieldType allows the storage and retrieval of a single file. It is capable of handling virtually any file type and is typically used for storing legacy document types such as PDF files, Word documents, spreadsheets, etc. The maximum allowed file size is determined by the "Max file size" class attribute edit parameter and the "upload_max_filesize" directive in the main PHP configuration file ("php.ini").

BinaryFile Value Object API

`eZ\Publish\Core\FieldType\BinaryFile\Value` offers the following properties.

Note that both `BinaryFile` and `Media` Value and Type inherit from the `BinaryBase` abstract field type, and share common properties.

| Attribute | Type | Description | Example |
|-----------------------|--------|---|---|
| <code>id</code> | string | Binary file identifier. This ID depends on the IO Handler that is being used. With the native, default handlers (<code>FileSystem</code> and <code>Legacy</code>), the ID is the file path, relative to the binary file storage root dir (<code>var/<vardir>/storage/original</code> by default). This attribute has been introduced in eZ Publish 5.2. | <code>application/63cd472dd7819da7b75e8e2fee507c68.pdf</code> |
| <code>fileName</code> | string | The human readable file name, as exposed to the outside. Used when sending the file for download in order to name the file. | <code>20130116_whitepaper_ezpublish5 light.pdf</code> |
| <code>fileSize</code> | int | File size, in bytes | <code>1077923</code> |
| <code>mimeType</code> | string | The file's mime type. | <code>application/pdf</code> |

| | | | |
|---------------|---------|--|---|
| uri | string | <p>The binary file's HTTP uri. If the URI doesn't include a host or protocol, it applies to the request domain. This attribute has been introduced in eZ Publish 5.2.</p> <p>The URI is not publicly readable, and must NOT be used to link to the file for download. Use <code>ez_render_field</code> to generate a valid link to the download controller.</p> <p>V5.4</p> <p>Since v5.4 the property contains the binary file's content/download URL.</p> | <p>/var/ezdemo_site/storage/original/application/63cd472dd7819da7b75e8e2fee507c68.pdf</p> <p>V5.4 /content/download/210/2707</p> |
| downloadCount | integer | Number of times the file was downloaded | 0 |
| path | string | <p>*deprecated* Renamed to <code>id</code> starting from eZ Publish 5.2. Can still be used, but it is recommended not to use it anymore as it will be removed.</p> | |

BinaryFile hash format

The hash format mostly matches the value object. It has the following keys:

- `id`
- `path` (for backwards compatibility)
- `fileName`
- `fileSize`
- `mimeType`
- `uri`
- `downloadCount`

REST API specifics

Used in the REST API, a BinaryFile field will mostly serialize the hash described above. However there are a couple specifics worth mentioning.

Reading content: url property

When reading the contents of a field of this type, an extra key is added: `url`. This key gives you the absolute file URL, protocol and host included.

Example: http://example.com/var/ezdemo_site/storage/original/application/63cd472dd7819da7b75e8e2fee507c68.pdf

Creating content: data property

When creating BinaryFile content with the REST API, it is possible to provide data as a base64 encoded string, using the `"data"` fieldValue key:

```
<field>
  <fieldDefinitionIdentifier>file</fieldDefinitionIdentifier>
  <languageCode>eng-GB</languageCode>
  <fieldValue>
    <value key="fileName">My file.pdf</value>
    <value key="fileSize">17589</value>
    <value
key="data"><![CDATA[/9j/4AAQSkZJRgABAQEAAZABkAAD/2wBDAAIBAQIBAQICAgICAgICAwUDAwMDAwYEBAMFBwYHBwCG
...
...]]></value>
  </fieldValue>
</field>
```