Image4IO API Specification Doc

API Version	Doc Version	Date	Author	Description	In-use
v0.0.1	-	26-July-2018	Image4IO API Team	Not publicly available	No
v0.0.2	-	28-July-2018	Image4IO API Team	Not publicly available	No
v0.0.3	v0.0.1	03-Aug-2018	Image4IO API Team	The first documentation for public alpha testing	No
v.0.0.3	v0.0.2	09-Aug-2018	Image4IO API Team	Usage has been added	No
v0.1.1		22-Apr-2019	Image4io API Team	All API endpoint renewed.	No
v0.1.2		02-May-2019	Image4io API Team	Folder/path system added.	No
v0.1.3		01-Jul-2019	Image4io API Team	Minor Fixes.	No
v0.1.4		26-Aug-2019	Image4io API Team	Minor Fixes.	Yes

Usage

Format

The requests are made in terms of a link. The format is:

- https://api.image4.io/{version}/{request}

API versioning format is "**vMAJOR.MINOR.PATCH**". In API url use only "**vMAJOR.MINOR**". An example of requesting to **UPLOAD** on version v0.1.1 would look like

- https://api.image4.io/v0.1/upload

Authentication

`HTTP Basic Authentication`
HTTP Basic Authentication needs to be used with every API request. API key should be used as `username` and API Secret should be used as `password`.

Standard Error Responses

Following response may be returned from any API call.

Status	Response

401	{
	"statusCode": 401,
	"message":"You are not authorized to do this operation."
	}

401	{
	"message": "User is not authorized to access this resource with an explicit deny"
	}

500	{
	"statusCode": 500,
	"message":"Internal Error"
	}

Methods

Upload

Upload a chosen file or multiple files to a specified directory

Request

Method	URL
`POST`	`{version}/upload`

Type	Params	Values
POST	multipart/form-data	file(s)
GET	Path	String
POST	usefilename	Bool
POST	overwrite	Bool

`multipart/form-data`
`multipart/form-data` is the requestor's file and must be sent with all `Upload` requests. Length must be between 3 and 20 characters. The file must have one of the following extensions: **.jpg**, **.png**, **.gif**.
After uploading successfully, the file is assigned a unique id.

`path` `[optional]`
`path` is the full path and specifies the directory where the file will be stored. Length must be between 3 and 20 characters. If not specified, it defaults to the user's root folder.

`usefilename` `[optional]`

`overwrite` `[optional]`

Response

Status	Response

200	**Response will be an object containing a list of the details of the file(s) as well as additional information. Each file has the following structure.**
	`{`
	`"original_name": <original name of uploaded file>,`
	`"name": <name of the file with assigned id>,`
	`"status": <status response>`
	`}`
	
	An example response could be:
	
	`[`
	`"uploadedFiles": [`
	`{`
	`"original_name": "sample1.jpg",`
	`"name": "scdbaf52e-5f32-4c69-8592-568037be859b.jpg",`
	`"status": "Uploaded"`
	`},`
	`{`
	`"original_name": "sample2.jpg",`
	`"name": "06ec5218-8fc1-4139-8488-622dbc045168.jpg",`
	`"status": "Uploaded"`
	`}`
	`]`
	`]`

400	`{`
	`"statusCode": 400,`
	`"message":"There are no files to upload."`
	`}`

401	`{`
	`"statusCode": 401,`
	`"message":"You are not authorized to do this operation."`
	`}`

500	`{`
	`"statusCode": 500,`
	`"message":"Internal error"`
	`}`

Get

Get information about an image

Request

Method	URL
GET	{version}/get

Type	Params	Values
GET	name	string

name
Name of file.

Response

Status	Response

200	**Response will be an object containing a list of the details of the file(s) as well as additional information. Each file has the following structure.**
	{
	"name": <name of file>,
	"userGivenName": <userGivenName>,
	"size": <size of file>,
	"format": <format of file>,
	"width": <width of file>,
	"height": <height of file>,
	"createdAtUTC": <created time>,
	"updatedAtUTC": <updated time>
	}
	
	An example response could be:
	
	{
	"name": "image.jpg",

	"userGivenName": "image2.jpg", "size": 2749, "format": "png", "width": 200, "Height":67, "createdAtUTC":"2019-04-20T12:28:21.133", "updatedAtUTC": "2019-04-20T12:28:21.133" }

400	{
	"statusCode": 400,
	"Message":"name parameter is required."
	}

401	{
	"statusCode": 401,
	"message":"You are not authorized to do this operation."
	}

500	{
	"statusCode": 500,
	"message":"Internal error"
	}

Copy

Copy a file with a new name

Request

Method	URL
PUT	{version}/copy

Type	Params	Values
GET	source	string
GET	target_path	string

source
`source` is the file which is going to be copied.

Target_path [optional]
`target_path` is the directory for copy of source image. Length must be between 3 and 20 characters and must include extension. Default value is root.

Response

Status	Response

200	**Response will be an object containing details of the copied file as well as additional information. The object has the following structure:**
	
	{
	"copiedFile": {
	"name": <name of the new file with assigned id>,
	"status": <status response>
	}
	}
	
	An example response could be:
	{
	"copiedFile": {
	"name": "dir/677c7737-257b-41b6-befd-692f3c4ca26d.jpg",

	"status": "Copied" } }

400	{
	"statusCode": 400,
	"message":"'source' parameter is required."
	}

400	{
	"statusCode": 400,
	"message":"There is no file with that name."
	}

401	{
	"statusCode": 401,
	"message":"You are not authorized to do this operation."
	}

500	{
	"statusCode": 500,
	"message":"Internal error"
	}

Move

Move a file to a different directory

Request

Method	URL
PUT	{version}/move

Type	Params	Values
GET	source	string
GET	target_path	string

source
`source` is the file which is going to be moved.

Target_path [optional]
`target_path` is the directory for copy of source image. Length must be between 3 and 20 characters and must include extension. Default value is root.

Response

Status	Response

200	**Response will be an object containing details of the moved file as well as additional information. The object has the following structure:**
	
	{
	"movedFile": {
	"name": <name of the new file with assigned id>,
	"status": <status response>
	}
	}
	
	An example response could be:
	{
	"movedFile": {
	"name": "dir/677c7737-257b-41b6-befd-692f3c4ca26d.jpg",

	<pre>"status": "Moved"
	}
	}</pre>

400	<pre>{
	"statusCode": 400,
	"message":"'There is no file with that name."
	}</pre>

401	<pre>{
	"statusCode": 401,
	"message":"'Source parameter is required."
	}</pre>

500	<pre>{
	"statusCode": 500,
	"message":"Internal error"
	}</pre>

Delete

Delete a file

Request

Method	URL
DEL	{version}/deletefile

Type	Params	Values
GET	name	string

name
The file that is to be removed.

Response

Status	Response
200	**Response will be an object containing details of the deleted file as well** }

400	{
	"statusCode": 400,
	"message":"'source' parameter is required."
	}

400	{
	"statusCode": 400,
	"message":"There is no file with that name."
	}

Status	Response
200	**Response will be an object containing details of the deleted file as well as**

	additional information. The object has the following structure:
	
	{
	  "deletedFile": {
	  "name": <name of the new file with assigned id>,
	  "status": <status response>
	  }
	}
	
	An example response could be:
	{
	  "deletedFile": {
	  "name": "dir/677c7737-257b-41b6-befd-692f3c4ca26d.jpg",
	  "status": "Deleted"
	  }
	}

400	{
	  "statusCode": 400
	  "message":"There is no file with that name."
	}

401	{
	  "statusCode": 401,
	  "message":"'name' parameter is required."
	}

401	{
	  "statusCode": 401,
	  "message":"You are not authorized to do this operation."
	}

500	{
	  "statusCode": 500,
	  "message":"Internal error."
	}

Delete Folder

Delete a folder.

Request

Method	URL
DELETE	{version}/deletefolder

Type	Params	Values
GET	path	string

path
The folder that is to be removed.

Response

Status	Response

200	**Response will be an object containing details of the fetched file as well as additional information. The object has the following structure:**
	
	{
	"deletedFolder": {
	"name": <name of the deleted folder>,
	"status": <status response>
	}
	}
	
	An example response could be:
	{
	"deletedFolder": {
	"name": "/folder",
	"status": "deleted"
	}

400	{
	"statusCode": 400,
	"message":"Path is need to be specified."
	}

401	{
	"statusCode": 401,
	"message":"You are not authorized to do this operation."
	}

404	{
	"statusCode": 404,
	"message":Folder is not found."
	}

500	{
	"statusCode": 500,
	"message":"Internal error."
	}

Fetch

Fetch a source from a link

Request

Method	URL
POST	{version}/fetch

Type	Params	Values
GET GET	from target_path	string string

from
The url that will be fetched `from`.

folder

`target_path` is the directory for copy of source image. Length must be between 3 and 20 characters and must include extension.

Response

Status	Response

200	**Response will be an object containing details of the fetched file as well as additional information. The object has the following structure:**
	
	{
	"fetchedFile": {
	"name": <name of the new file with assigned id>,
	"status": <status response>
	}
	}
	
	An example response could be:
	{
	"fetchedFile": {
	"name": "dir/677c7737-257b-41b6-befd-692f3c4ca26d.jpg",
	"status": "Fetched"
	}
	}

400	{
	"statusCode": 400,
	"message":"File source is invalid."
	}

400	{
	"statusCode": 400,
	"message":"Not supported file type."
	}

400	{
	"statusCode": 400,
	"message":"There is no file with that name. "
	}

401	{ "statusCode": 401,

	<pre>"message":"'from' is not fully qualified url."
	}</pre>

401	<pre>{
	"statusCode": 401,
	"message":"'from' parameter is required."
	}</pre>

500	<pre>{
	"statusCode": 500,
	"message":"Internal error."
	}</pre>

Create Folder

Create a new folder

Request

Method	URL
POST	{version}/path

Type	Params	Values
GET	path	string

path
Name of new folder.

Response

400	<pre>{
	"statusCode": 400,
	"Message":"name parameter is required."
	}</pre>

401	<pre>{
	"statusCode": 401,
	"message":"You are not authorized to do this operation."
	}</pre>

500	<pre>{
	"statusCode": 500,
	"message":"Internal error"
	}</pre>

Status	Response

200	**Response will be an object containing a list of the details of the file(s) as well a additional information. Each file has the following structure.**
	
	{
	"createdFolder": {
	"Name":"new_folder",
	"Status":"created",
	}
	}

401	{
	"statusCode":,
	"message":"You are not authorized to do this operation."
	}

500	{
	"statusCode": 500,
	"message":"Internal error"
	}

List Folder

List all stored files.

Request

Method	URL
GET	{version}/listfolder

Type	Params	Values
GET	path	string

Response

Status	Response

200	**Response will be an object containing details of the fetched file as well as additional information. The object has the following structure:**
	
	{
	"folders": [
	{
	"name": <name of the folder>,
	"parent": <name of parent folder>
	}
	],
	"files": [
	{
	"orginal_name": <original name of the file>,
	"name": <name of the file with assigned id>,
	"size": <size of file in bytes>,
	"format": <format of image>,
	"width": <width of image>,
	"height": <height of image>,
	"createdAt": <creation date of file in UTC>,
	"updatedAt": <update date of file in UTC>
	}

	]
	}
	
	An example response could be:
	{
	"folders": [
	{
	"orginal_name": "testfolder",
	"name": "testparent",
	}
	],
	"files": [
	{
	"orginal_name": "tesstst",
	"name": "6f3e2ad3-00da-490c-a8d2-e3f1a372.jpg",
	"size": 186677,
	"format": "jpg",
	"width": 975,
	"height": 1300,
	"createdAt": "2019-04-20T12:27:31.48",
	"updatedAt": "2019-04-20T12:27:31.48"
	},
	{
	"orginal_name": "tesstst",
	"name": "e7087a36-e6cc-432e-bb67-c57b42a9e.jpg",
	"size": 186677,
	"format": "jpg",
	"width": 975,
	"height": 1300,
	"createdAt": "2019-04-20T12:28:21.133",
	"updatedAt": "2019-04-20T12:28:21.133"
	}
	]
	}

401	{
	"statusCode": 401,
	"message":"You are not authorized to d1 this operation."
	}

404	{ "statusCode": 404, "message":"Folder is not found."

	}

500	{
	"statusCode": 500,
	"message":"Internal error."
	}

Transformation

URL

`https://cdn.image4.io/{cloudname}/{image-folder-and-name}`

URL structure is shown above.

Parameters

`Width | w`

The width of the resulting picture is denoted as `w`. If a transformation takes only one input, either `h` or `w`, but both are entered, it will take `w` as standard.

`Height | h`

The height of the resulting picture is denoted as `h`. If a transformation takes only one input but both are entered, it will disregard `h` and will take `w` instead.

`Fit Mode | fit`

The style of transformation that will be done to the image. Two modes possible:

- `saveratio` - saves the width-to-height ratio of the resulting image. Takes either `w` or `h`.
- `crop` - cuts off the edges. Takes both `w` and `h`. If only one parameter is specified, the cropped image will adjust the other to keep the width-to-height ratio of the original image.

If none specified, it defaults to `saveratio`.

`Maximum Width, Minimum Width | maxw, minw`

The maximum or minimum value `w` will be adjusted to if `w` is not specified. Only applicable if `fit` mode is set as `saveratio`.

`Maximum Height, Minimum Height | maxh, minh`

The maximum or minimum value `h` will be adjusted to if `h` is not specified. Only applicable if `fit` mode is set as `saveratio`.

`Crop Region | c`

The specified region the image will be cropped in is denoted in `c`. Only applicable if `fit` mode is set as `crop`. The region can be denoted as:

- `top, left`
- `top, right`
- `bottom, left`
- `bottom, right`
- `center`

If no valid region is entered, it defaults to `top, left`.

Format | f
The format of the resulting image is denoted in `f`. Formats must be one of the following:

- gif
- png
- jpeg

If none or different formats are entered, it defaults to jpeg.

Note: Transforming a gif image will always result in jpeg encoding and `Content-Type: image/jpeg` header.

Quality | q
The resulting quality of the image. Must be a value between 0 and 100. If entered outside of range or not specified, it defaults to 75.

Important Notice for Alpha Testing

Server Cold Start Time

In our alpha testing server, there is a latency for the first call to the API after a long period of time. Due to having few test users you may experience this latency. After the first successful call, however, successive responses will be more quickly.

Glossary

Conventions

- **Client** - Client application.
- **Status** - HTTP status code of response.
- All the possible responses are listed under ‘Responses’ for each method. Only one of them is issued per request server.
- All response are in JSON format.
- All request parameters are mandatory unless explicitly marked as `[optional]`

Status Codes

All status codes are standard HTTP status codes. The below ones are used in this API.

`2XX` - Success of some kind
`4XX` - Error occurred in client’s part
`5XX` - Error occurred in server’s part

Status Code	Description
`200`	`OK`
`400`	`Bad request`
`401`	`Authentication failure`
`426`	`Upgrade required`
`500`	`Internal server error`