API Documentation

Except of the Upload URLs (Step 2 & 2a) and the Thumbnail URLs, that are not part of the API but "live" independent, all other method requests are sent to the URL **https://devs.sparevideos.com/api/v1** followed by the method's **/endpoint** as indicated hereafter.

All requests must be signed and the signature (token) must be sent to the url as a GET/POST variable like: **/endpoint?token={signature}**. The signature is composed by 2 parts, separated by a dot (.)

- The first part is a base64URLEncode array, without the final equal signs of base64 encode (==)
- The second part is an HMAC sha256 hash with the array just mentioned and the {private_key} as "salt"

For example to initiate an upload request in PHP, you should create an array:

```
$upload_init = [
'algo'        =>    'sha256',
'public_key'  =>    '{you_public_key}',
'method'      =>    'upload',
'action'      =>    'initiate',
'expires'     =>    time()+(60*60)
];
$base64Encode = trim(base64_encode(json_encode($upload_init)),'=');
```

The hash would be:

```
$encryptedHash = hash_hmac('sha256', $base64Encode,{your_private_key})
```

The signature (token) would be:

```
$signature = $base64Encode. '.' . $encryptedHash
```

Result looks like:

VGhpcyBpcyBhbiBlbmNvZGVkIHN0cmluZw.b8e7ae12510bdfb1812e463a7f086122cf37e4f7

The signature asks the API for permission to upload a file within 1 hour (expires: time()+(60*60)). Now you just need to request:

https://devs.sparevideos.com/api/v1/upload/initiate?token=$signature

That would return a JSON Encoded array with information about the upload. For example:


```
{ "status": "success",
  "server": "https://www.sparevideos.com/fileserver/upload",
  "ref": "1487800240997580 2658ae07b05035c",
  "token":
"eyJhbGdvIjoic2hhMjU2IiwibWV0aoidXBsbw0aW9uIjoiwicHVtleSI6IjM0NlUzNjg1RiZWZhYWU3MTk4IiwiZXhwaXTQ4NzgwMzg0MvdXBsb2FkIiwicmVmIjoiMTQ4NzgwMDI0MDk5NzU4MDI2NThhZTA3YjA1MDM1YyIsInRpbWUiOjEMDAyNDB9fQ.b98547ersdffe84bb71aafe1e3f8d960ecb540856cd112965rgd99060aab26a"
}
```


This gives you the URL where you need to upload the file to and additional variables to pass to the request.

Whatever is in bruckets {} in the documentation below, take it as a variable which is used in other parts of the documenation.

Upload process

Step 1: Make upload initiation request

This method will return to us the available server details to which we will upload our file.

Endpoint: **/upload/initiate**
HTTP Method: **GET**

	Parameter	Value	Description
Signature Array:	algo method action public_key expires	"sha256" "upload" "initiate" {public_key} #seconds UNIX Time	**See note below the table*
Response:	status	"success" or "error"	
If "success":	server ref token	**{upload_url}** **{ref_number}** **{upload_token}**	*Notice we now receive a new token. This is not the same we previously generated but the one we received in the JSON response.*
If "error":	error message	numeric value user readable string	

**See available encryption algos: http://php.net/manual/es/function.hash-algos.php*

Step 2: Upload the file

This is not a method of the API. This is where you will actually send the file. As you see the endpoint is the upload_url we received in the response of the Upload/Initiate API as "server" and the token as "token".

Endpoint: **{upload_url}?token={upload_token}**
HTTP Method: **POST**

	Parameter	Value	Description
Request:	remote_file *- or -* file	URL to file Local file	*Only 1 of two must be sent*
Response:	status	"success" or "error"	
If "success":	code	{file_code}	
If "error":	error message	numeric value user readable string	

Step 2a: Track file upload progress (Optional for remote file uploads only)

When you upload a file from your computer/machine you can track the upload progress as you know how many bytes have been uploaded at any point of time. If you just send a request to fetch a file from a remote url though you can't know the progress. So what you need to do is to call the endpoint below at the frequency you wish and get a plain text numeric value which will tell you the progress out of 100 of your upload.

Endpoint: **{upload_url}/progress**
HTTP Method: **GET**

	Parameter	Value	Description
Request:	ref	**{ref_number}**	*As retrieved by /upload/initiate*
Response:	plain text, numeric value out of 100		

Step 3: Request conversion

Sometimes, after you upload a file and we return you the file code, for some reason you may not be able to save it in your DB. To prevent that the file gets converted and counts as stored and converted (thus produce a cost for you), you need to store the file on your DB and then request conversion when you are ready. Please note 2 and 2a call external URLs, while now we are back to our API's endpoints. Also note that the token we use as GET variable is the login_token which you need to regenarate if it is expired by that time.

Endpoint: **/upload/conversion/initiate**
HTTP Method: **GET**

	Parameter	Value	Description
Signature Array:	algo	"sha256"	
	method	"upload"	
	action	"convert"	
	public_key	{public_key}	
	file_code	{file_code}	
	expires	#seconds UNIX Time	
	convert_sd_only	True|False	*Optional, Default:False*
	convert_mp4_only	True|False	*Optional, Default:False*
Response:	status	"success" or "error"	
If "success":	code	The file code	
If "error":	error	numeric value	
	message	user readable string	

File

Retrieve file information

With this method you can get file information, thumbnail, duration, file type, available formats' urls if it is converted and ready to be streamed etc...

Endpoint: **/me/file**
HTTP Method: **GET**

	Parameter	Value	Description
Signature Array:	algo	"sha256"	
	method	"file"	
	action	"get"	
	public_key	{public_key}	
	file_code	{file_code}	
	expires	#seconds UNIX Time	
Response:	status	"success" or "error"	
If "success":	extension	mp4, webm, avi, mov etc...	*original file extension*
	playtime	numeric value in seconds	
	thumbnail	Full image URL	
	streams	array()	
If "error":	error	numeric value	
	message	user readable string	

The parameter "streams" is an array with the following keys: hls, mpd, mp4 and, if available, also webm. "hls" and "mpd" are arrays with file=>url key-value pair, while mp4 and webm return also arrays but with many subarrays, as many as the available formats, which have file=>url and label=>number key-value pairs.

An example will make it easier:

```
{   "status": "success",   "extension": "avi", "playtime": 15, "thumbnail": " https://url-to-thumbnail ",
 "streams": {
  "mp4": [
   {
    "file": "https://url-to-file ",
    "label": "360"
   },
   {
    "file": " https://url-to-file ",
    "label": "720"
   }
  ],
  "hls": {
   "file": " https://url-to-file "
  },
  "mpd": ...
 }}
```

Update file information

With this method you can update the file thumbnail

Endpoint: **/me/file**
HTTP Method: **PUT (POST with input "_method"="UPDATE")**

	Parameter	Value	Description
Signature Array:	algo	"sha256"	
	method	"file"	
	action	"update"	
	public_key	{public_key}	
	file_code	{file_code}	
	thumbnail	{thumbnail_file_name}	*See Thumbnail LIST*
	expires	#seconds UNIX Time	
Response:	status	"success" or "error"	
If "error":	error	numeric value	
	message	user readable string	

File Thumbnail(s)

We have several methods under https://static.sparevideos.com/thumbnail that help list all available thumbnails as well as return a specific one. All these methods do not require any signature.

/{file_code}/json/list

- This path will return a JSON list of all available thumbnails
- The keys of the list represent the height and the sequential number of the image separated by a dot
- The dot separator allows numeric sort by keys correctly
- The values are the exact image file names as you will need to pass to File/Update method

example: /1488633857-1lPYNC/json/list
result:{"720.05":"1488633857-1lPYNC_720_05.jpg","720.04":"1488633857-1lPYNC_720_04.jpg","720.03.selected":"1488633857-1lPYNC_720_03.jpg","720.02":"1488633857-1lPYNC_720_02.jpg","720.01":"1488633857-1lPYNC_720_01.jpg","480.05":"1488633857-1lPYNC_480_05.jpg","480.04":"1488633857-1lPYNC_480_04.jpg","480.03":"1488633857-1lPYNC_480_03.jpg","480.02":"1488633857-1lPYNC_480_02.jpg","480.01":"1488633857-1lPYNC_480_01.jpg","360.05":"1488633857-1lPYNC_360_05.jpg","360.04":"1488633857-1lPYNC_360_04.jpg","360.03":"1488633857-1lPYNC_360_03.jpg","360.02":"1488633857-1lPYNC_360_02.jpg","360.01":"1488633857-1lPYNC_360_01.jpg"}

/{file_code}/exact/{file_name}

- This path will return an image
- The file_name is the value of one of the key=>value items in the JSON list above
- Note 720_03_selected has _selected appended, this means you have updated the file thumbnail with that key's value

example: /1488633857-1lPYNC/exact/1488633857-1lPYNC_720_02.jpg

/{file_code}/{size}/default.jpg
/{file_code}/{size}/{file_name}

- This will return the default or selected image resizing it to the indicated size
- The size must be a number in pixels (e.g. 300)
- If the default image is less than 260 it will return 260h image

example: /1488633857-1lPYNC/300/default.jpg
example: /1488633857-1lPYNC/300/1488633857-1lPYNC_720_02.jpg

Files

Retrieve all the files you have uploaded

Paginated with a maximum of 2000 file codes per page.

Endpoint: **/me/files**
HTTP Method: **GET**

	Parameter	Value	Description
Signature Array:	algo	"sha256"	*Required*
	method	"files"	*Required*
	action	"get"	*Required*
	public_key	{public_key}	*Required*
	expires	#seconds UNIX Time	
Response:	status	"success" or "error"	
If "success":	See example below		
If "error":	error	numeric value	
	message	user readable string	

```
{   "status": "success",   "currentPage": 1,  lastPage": 3,
  "nextPageUrl": "https://url-to-page2",
  "codes": [
    "1484846059-40SVCC",
    "1484906398-Il6ssC",
     ...
  ]
}
```

Note: If you use nextPageUrl to go to next page each time, remember to postpone &token={login_token} so that it authorizes the action.

Advices:

- When you sign to create an upload use an expiration that would be enough for the upload to start by the user. You can use even 1 or 2 hours if you want.
- When you create a signature for File/Get or File/Update API to retrieve or update file info (stream URLs are included in the returned result), you should use a short expiration time like 10 to 30 seconds that are usually enough for the API to receive your request, execute it and give you back your results.

Delete File(s)

When you want to delete files, you just need to prepare the files' codes you want to delete appropriately and pass them in the array to sign.

Endpoint: **/me/files**
HTTP Method: **DELETE (POST with input "_method"="DELETE")**

	Parameter	Value	Description
Signature Array:	algo	"sha256"	
	method	"files"	
	action	"delete"	
	public_key	{public_key}	
	expires	#seconds UNIX Time	
	codes	array()	
Response:	status	"success" or "error"	
If "success":			
If "error":	error	numeric value	
	message	user readable string	

Error codes:

401	Not authorized
403	Forbidden
404	Not found
500	Internal Error
501	Service not available
1000	Generic error

File upload specific errors:

10000	Upload not executed
10001	Fileserver process error
10002	Upload not valid
100030	File not converted
100031	File already converted
10004	Internal system error
10005	Service availability error
10006	Service limits exceeded