Animate 3D REST API

Revisions

Alpha v1.0.0

Initial rest APIs

Alpha v1.0.1

Added model parameter to the /process API

Alpha v1.2.0

Added custom character end points /character

Alpha v1.2.1

Added the “sim” parameter to API 3: Start Video Processing

Alpha v1.2.2

Added the “camera” parameter to API 3: Start Video Processing

Alpha v1.3.0

Added webhook APIs

Alpha v1.4.0

Added footLockingMode parameter to the /process API
Added new minutesBalance API

Alpha v1.4.1

Added flag “createThumb” to /character/storeModel API

Alpha v1.5.0

Exposed mp4 render out parameters in /process API
Added Face Tracking parameter in /process API

Alpha v1.5.1

Added /videoInfo API
Added videoSpeedMultiplier, poseFilteringStrength, rootAtOrigin parameters & new mp4 options in /process API

Alpha v1.5.2

Added /character/deleteModel API

Alpha v1.5.3

Added trim & crop parameters in /process API

Alpha v1.5.4

Added /account/creditBalance API

Alpha v1.5.5

Added Hand/Finger Tracking parameter in /process API

Alpha v1.5.6

Added parameter for rerunning a job in /process API. API Error codes with information have been added.

Alpha v1.5.7

Added error messages

Alpha v1.5.8

Added stockModel query param in /listModels api

Alpha v2.0.0

Added experimental multi person api

The Animate 3D REST API lets you convert videos into 3D animations without having to use the DeepMotion Web Portal. Instead you can upload, process, and download the resulting FBX/BVH animations directly from an external application like a web or desktop app.

Authentication

The Animate 3D REST API uses basic **HTTP Authentication** to keep your requests and data secure. To use the API you will need a **Client ID** and a **Client Secret** which are provided by DeepMotion. If you do not have these please contact DeepMotion Support or your sales representative.

To retrieve your API access token you need to add the following Authorization header to your token request:

```
Authorization: Basic Base64(<clientId>:<clientSecret>)
```

where the value of `<clientId>:<clientSecret>` is **base 64** encoded. For Example, if your Client ID is `1a2b` and your client Secret is `3c4d` then your authorization header should look like this:

```
Authorization: Basic MWEyYjozYzRk
```

where `MWEyYjozYzRk` is the base64 encoded value of `1a2b:3c4d`.

API Endpoints

All Animate 3D API requests must be made against the following base URL using the HTTPS protocol and port:

```
Production Environment:     (Contact DeepMotion)
```

For using our API from browser javascript locally (to avoid CORS error), please send request from any of the origin below:
http://localhost:8080
http://localhost:8180
For production deployment, please let us know your production url (scheme, host, port), so that we can configure our CORS setting accordingly.

API Reference

API 1: Get Access Token

Desc	Authenticate client credentials and returns a time limited session cookie to be used in the subsequent REST API calls. After the session expiration, this API needs to be called again to get a new session cookie
Method + URI	GET {host}/session/auth
Header(s)	Authorization: Basic Base64(<clientId>:<clientSecret>)
Request	
Response	Sample Response Header: set-cookie: dmsess=s%3AEsF23MoyDEq7tTWQM8KfA_wjKkSrOFwU.2fjJTfDP%2FT2BeA5DFenwOH4t8XzqZsbSc6M2mZwS%2BWg; Domain=.deepmotion.com; Path=/; Expires=Mon, 03 Aug 2020 13:36:26 GMT; HttpOnly (Note: **dmsess** is the session cookie. This cookie needs to be sent in all subsequent REST API calls. Sample Request Header for other API calls: cookie:dmsess=s%3AEsF23MoyDEq7tTWQM8KfA_wjKkSrOFwU.2fjJTfDP%2FT2BeA5DFenwOH4t8XzqZsbSc6M2mZwS%2BWg)

API 2: Upload Video

Desc	Retrieves a signed url to upload video
Method + URI	GET {host}/upload
Header(s)	cookie:dmsess=<cookie-value-returned-from-authentication-api>
Request	Query parameters: <name>: video/image file name with extension (like test.mp4 or test.jpg) <resumable>: 0 or 1(default) returns resumable or regular signed url (optional)

Response	JSON object:
	{
	  "url": signed url
	}
	
	After retrieving the url, actual video upload is required to that storage url. If ’resumable’ option is set in the request, we need one POST and one subsequent PUT request, otherwise a single PUT request will do the job.
	
	POST request to url:
	<x-goog-resumable>: start (set in the request header)
	<location>: resumable url (set in the response header by server)
	
	Put request to resumable url/url:
	attach raw bytes of the video file in the request body.

API 3: Start Video Processing

Desc	Start processing video after file has been uploaded to the designated URL
Method + URI	POST {host}/process
Header(s)	cookie:dmsess=<cookie-value-returned-from-authentication-api>

Request	POST body should include a JSON object:
	{
	  “url”: <upload url>
	  “rid”: <previous successful job’s request id>
	  “processor”: <processor_id>
	  “params": [<params>, ...]
	}

<upload_url> should match url returned from GET /upload request.

To rerun a job with different parameters, “rid” input should be used instead of “url”.

<processor_id> specifies which processor to use to process the video file, must be one of the following:

Processor Id	Description
video2anim	Deepmotion video to animation processor

<params> specifies additional parameters that will be passed to the specified processor, for example:
   "params": [
"config=configDefault",formats=bvh,fbx,mp4,model=<modelId>]

For static pose, png/jpg can be included in the formats parameter, like: formats=bvh,fbx,png (to output rendered image instead of rendered video)

Additional important parameter: **sim**
This physics simulation parameter needs more clarification. This parameter influences Pose Estimation result to improve it in some cases like body parts inter penetration etc. If we would like to turn this ON, add sim=1 OR add sim=0 to turn it OFF. If we don’t add this parameter, simulation is turned off by default.

Added face tracking support:
trackFace

- Enable tracking basic facial expressions. Compatible with character models that contain ARKIT blend shapes. Enabling this option increases animation processing time (and additional charge if the server processes this for a compatible model).
- Default value is 0 and value can be either 0 or 1

Added hand tracking support:
trackHand

- Enable tracking hand/finger movement. Compatible with character models that contain hand/finger joints. Enabling this option increases animation processing time (and additional charge if the server processes this for a compatible model).
- Default value is 0 and value can be either 0 or 1

Another new parameter is: **poseEstimation.footLockingMode** or simply **footLockingMode**

- This parameter value can be one of the below:
    - **auto** : default mode, automatic switching between locking and gliding modes of the foot, recommended for general cases
    - **always** : forced foot locking all the time. only used when Auto mode can not remove all the foot gliding unsired
    - **never** : forced to disable foot locking and character grounding. used when the motion is completely in the air or in the water and therefore neither foot locking nor character grounding is needed.
    - **grounding** : forced disabling foot locking, however character is still grounded. Only used when Auto mode prevents the desired foot gliding (i.e. during shuffling dances) in the motion or locks the foot for too long on the ground during fast and short foot/ground contacts (i.e. during sprints or jumps.)

We have added few new parameters for better body tracking results:

poseEstimation.videoSpeedMultiplier or simply **videoSpeedMultiplier**

- For input videos that have been slowed down, enabling this option can help improve the resulting animation quality. For example, if your input video speed moves at 1/2 speed, then set the speed multiplier to 2x to improve animation quality
- Default value is 1.0 and range is 1.0 - 8.0

poseEstimation.poseFilteringStrength or simply **poseFilteringStrength**

- Applies an advanced AI filter that helps remove jitter and produce smoother animations though may result in lower animation accuracy for certain frames or sequences
- Default value is 0.0 and range is 0.0 - 1.0

rootAtOrigin

- Place a root joint at the origin of the output character. This is helpful in some cases, for example, for UE4 retargeting.
- Default value is 0 and value can be either 0 or 1

Trim (input video only) & Cropping(input video/image)

- trim=from,to (in seconds, example: trim=1,2.6)

	• crop=left,top,right,bottom (normalized coordinate value, origin[0,0] is left-top. Like, if original image is let's say [1080 x 1920], than applying the crop: `crop=0.239,0.121,0.742,0.981` would give us [543 x 1652]. **Mp4 render out parameters:** 1. Please add this below parameter, if you would like to generate the mp4 with only animated character in a default background (and without the original video): **render.sbs=0** 2. To replace the default background with a solid color (for green screening etc.) **render.sbs=0** **render.bgColor=0,177,64,0** (RGBA color code in the range of 0-255 for each channel, please note, the last channel (alpha) value is not in effect) 3. To set a studio like background with a solid color tint **render.sbs=0** **render.backdrop=studio** **render.bgColor=0,177,64,0** 4. To enable character shadow **render.shadow=1** 5. **render.includeAudio** • When enabled, it includes the audio of the original input video in the generated animation. • Default value is 1 and value can be either 0 or 1 6. **render.CamMode** values are below. Default is 0 0 (Cinematic) The character is kept in the center of the frame 1 (Fixed) Camera will stay fixed relative to the background 2 (Face) Camera keeps the torso and face in the center of frame

Response	JSON object:
	{
	  "rid": <request id>
	}

API 4: Poll for Job Status

Desc	Polls for real-time status of a given processing job
Method + URI	GET {host}/status/rid GET {host}/status/rid1,rid2,..,rid
Header(s)	cookie:dmsess=<cookie-value-returned-from-authentication-api>
Request	Clients can request current status of previously submitted processing requests (API3). Use comma (‘,’) to separate multiple request ids if retrieving status for more than 1 request.

Response	JSON object:
	{
	  "count": <number of records in status array>,
	  "status": [
	    <status>,
	    … …
	  ]
	}
	
	Each element in status array is a JSON object:
	{
	  "rid": <request id>,
	  "status": <status name>
	  "details": <status details, see below>
	}
	
	<status name> is one of the following **case sensitive** values:
	
	Status Name — **Description**
	PROGRESS — Request is still being processed
	SUCCESS — Request is processed successfully
	RETRY — Request has failed for some reason, but is being retried
	FAILURE — Request has failed
	
	<status details> for PROGRESS:
	{
	  “step”: <current step>,
	  “total”: <expected total number of steps>
	}
	
	<status details> for SUCCESS:

Status Name	Description
PROGRESS	Request is still being processed
SUCCESS	Request is processed successfully
RETRY	Request has failed for some reason, but is being retried
FAILURE	Request has failed

	{
	  “In”: <original video file>,
	  “out”: <processed video file>
	}
	
	<status details> for RETRY and FAILURE include last error message. Currently the format is:
	{
	  “exc_message”: <exception message, if any>,
	  “exc_type”: <exception type, if any>
	}
	But please note the format may change if we decide to mask error information (or pass more information) to client applications.

API 5: Get Download URLs

Desc	Get download URLs for the specified request ids
Method + URI	GET {host}/download/rid GET {host}/download/rid1,rid2,...,rid
Header(s)	cookie:dmsess=<cookie-value-returned-from-authentication-api>
Request	Clients can request download URLs for finished processing requests. Use comma (‘,’) to separate request ids if retrieving download URLs for multiple processing requests.

Response	JSON object:
	{
	  “count”: <number of records in links array>,
	  “links”: [
	    <link>,
	    … ...
	  ]
	}
	
	Each element in links array is a JSON object:
	{
	  “rid”: <request id>,
	  “name”: <name of the video>
	  “size”: <size of the video>
	  “duration”: <duration of the video>
	  “input”: <link of the video>
	  “urls”: [
	   {
	    “name”: <name of the downloadable item>

	“files”: <links of the files by extension> [
	{ <file type>: <URL to download the corresponding file>},
	{<file type>: <URL to download the corresponding file>}
	]
	}
	]
	}
	
	For example, if a processor outputs both bvh and fbx files, then the download link object will look like:
	{
	“rid”: “1234567890”,
	“urls”:[{
	“files”: [
	{“bvh”: “https://.../…”},
	{“fbx”: “https://.../…”}
	]
	}]
	}
	
	Please note that if the specified request has not finished yet or has failed, the response will not include any download urls, and the link object will look like:
	{
	“rid”: “1234567890”
	}
	
	Note: .dmpe format is available with the name **landmarks.dmpe**

API 6: List All Video Processing requests by Status

Desc	List past and current request ids Note: failed jobs and old jobs may be removed by system after a predefined retention period
Method + URI	GET {host}/list GET {host}/list/status1,...,status
Header(s)	cookie:dmsess=<cookie-value-returned-from-authentication-api>
Request	Client can request to get list of existing request ids of current user Client can specify one or multiple status value(s) to retrieve only request ids with the same status value(s). For example, GET

	/list/PROGRESS will only return list of requests that are still being processed

Response

JSON object:

```
{
   "count": <number of records in the rids array>,
  "list": [
    {
      "rid": "1234567890-1234567890-1234",
      "fileName":"",
      "fileSize":0,
      "fileDuration":0,
      "status": "PROGRESS",
      "ctime": <creation time>,
      "mtime": <last modification time>
    },
    ... ...
  ]
}
```

Each element in list is a JSON object with the following fields defined:

Field	**Description**
rid	Request/emoji id
fileName	Input video file name
fileSize	Input video file size in bytes
fileDuration	Input video duration in seconds
status	Current status (STARTING, PROGRESS, SUCCESS, FAILURE, RETRY)
ctime	Creation time (milliseconds since epoch)
mtime	Last modification time (milliseconds since epoch)

API 7: Credit Balance

Desc	Retrieves Credit Balance for an user
Method + URI	GET {host}/account/creditBalance

Header(s)	cookie:dmsess=<cookie-value-returned-from-authentication-api>
Request	n/a

Response	JSON object:
	{
	"credits":<value>
	}

API 8: Input Video Information

Desc	Get Video information such as resolution, fps etc.
Method + URI	POST {host}/videoInfo
Header(s)	cookie:dmsess=<cookie-value-returned-from-authentication-api>

Request	POST body should include a JSON object:
	{
	“url”: <upload url>
	}
	
	<upload_url> should match url returned from GET /upload request AND the video needs to be uploaded to that url in GCS before calling this API

Response	JSON object: {"width":1080,"height":1080,"fps":30,"duration":3,"codec":"h264","size":186615}

Custom Character APIs

Note: To make uploaded model(s) available to all animation jobs, please make sure x-useruid HTTP(S) header should have **not** been passed to the {host}/auth API to get the session for API 1 and API 2 below.

API 1: Model Upload Url

Desc	Retrieves signed urls to upload 3d model data(fbx, glb, gltf, or vrm format) and thumbnail(preferably png format)
Method + URI	GET {host}/character/getModelUploadUrl
Header(s)	cookie:dmsess=<cookie-value-returned-from-authentication-api>
Request	Query parameters: <name>: base name of the files (without extension) (optional) <modelExt>: file extension of the model file. Example: fbx (optional) <thumbExt>: file extension of the thumb file. Example: jpg (optional) <resumable>: 0 or 1(default) returns resumable or regular signed url (optional)

Response	JSON object:
	{
	"modelUrl": signed url
	"thumbUrl": signed url
	}
	
	After retrieving the urls, actual model & thumbnail upload are required to that storage urls. If 'resumable' option is set in the request, we need one POST and one subsequent PUT request for each signed url, otherwise a single PUT request will do the job per url.
	
	POST request to url:
	<x-goog-resumable>: start (set in the request header)
	<location>: resumable url (set in the response header by server)
	
	PUT request to resumable url location/url:
	attach raw bytes of the model or thumbnail file in the request body.

API 2: Store Model

Desc	Store the asset paths returned from getModelUploadUrl in database

Method + URI	POST {host}/character/storeModel
Header(s)	cookie:dmsess=<cookie-value-returned-from-authentication-api>
Request	Body parameters: <modelUrl>: model url returned from API 1 (optional if <modelId> is provided) <modelName>: model name (optional) <thumbUrl>: thumbnail url returned from API 1 (optional) <modelId>: model id to update existing model info (name or thumb) (optional if <modelUrl> is provided) <createThumb>: 0 (default) or 1, indicate if the thumbnail of the model needs to be generated (optional)

Response	JSON object:
	{
	  “modelId: Unique model id that can be passed to video process API
	}

API 3: List Models

Desc	List models based on specific query or without
Method + URI	GET {host}/character/listModels
Header(s)	cookie:dmsess=<cookie-value-returned-from-authentication-api>
Request	Query parameters: <modelId>: existing model id (optional) <searchToken>: for example search by model name (optional) <stockModel>: = When this parameter is supplied, all stock models (including deepmotion & roblox) will return in api response along with the account's custom models. Beside that, each model details now include a platform field which can be one of the below values : custom, deepmotion, roblox . (optional)
Response	JSON object: [{ “Id: Unique model id that can be passed to video process API “name”: name of the model “thumb”: url of the thumbnail if exist “rigId”: rigTemplate id with which this model is associated with

	“ctime”: creation timestamp “mtime”: modification timestamp “platform”: platform of the model }]

API 4: Delete Model

Desc	Delete model with specific model ID
Method + URI	DELETE {host}/character/deleteModel/<model ID>
Header(s)	cookie:dmsess=<cookie-value-returned-from-authentication-api>
Request	

Response	JSON object:
	{
	“count”: number of models that have been deleted
	}

Experimental Webhook APIs

Event Payload

Headers:

HTTP POST payloads that are delivered to your webhook's configured URL endpoint will contain the following headers:

X-DeepMotion-Signature: <signature>

Note: Signature is your client ID. It is supposed to be verified by your event handling code.

Body:

The event body is a JSON object described below:

```
{
        "eventType": <event type>,
        "data" <event data>
}
```

The following table explains the currently supported event types and their data sub-attributes. Data sub-attributes is also a JSON object:

eventType	Description	data	Note

job.completed	A task is completed	{
		"taskId": <request ID>
		“status”: <success|failure>
		}
			taskId is the rid that returned by the POST {host}/process API

API 1: Create a webhook endpoint

Desc	Create a webhook endpoint
Method + URI	POST {host}/webhook_endpoints
Header(s)	cookie:dmsess=<cookie-value-returned-from-authentication-api>

Request	JSON object:
	{
	"url": <endpoint URL>,
	"events": <array of events that would register with this endpoint>
	}

Response	JSON object:
	{
	"id": <endpoint ID>,
	"object": "webhook_endpoint",
	"url": <endpoint URL>,
	"events": <array of events that would register with this endpoint>
	}

API 2: Retrieve a webhook endpoint

Desc	Retrieve a webhook endpoint
Method + URI	GET {host}/webhook_endpoints/<endpoint ID>
Header(s)	cookie:dmsess=<cookie-value-returned-from-authentication-api>
Request	
Response	JSON object: Response: { "id": <endpoint ID>, "object": "webhook_endpoint", "url": <endpoint URL>,

	"events": <array of events that would register with this endpoint> }

API 3: List webhook endpoints

Desc	List webhook endpoints
Method + URI	GET {host}/webhook_endpoints
Header(s)	cookie:dmsess=<cookie-value-returned-from-authentication-api>
Request	
Response	JSON object: { "count": <number of endpoints>, "endpoints": [{ "id": <endpoint ID>, "object": "webhook_endpoint", "url": <endpoint URL>, "events": <array of events that would register with this endpoint> }, ...] }

API 4: Update a webhook endpoint

Desc	Update a webhook endpoint
Method + URI	POST {host}/webhook_endpoints/<endpoint ID>
Header(s)	cookie:dmsess=<cookie-value-returned-from-authentication-api>
Request	JSON object: { "url": <endpoint URL>, "events": <array of events that would register with this endpoint> }
Response	JSON object: { "id": <endpoint ID>, "object": "webhook_endpoint", "url": <endpoint URL>, "events": <array of events that would register with this endpoint>

	}

API 5: Delete a webhook endpoint

Desc	Delete a webhook endpoint
Method + URI	DELETE {host}/webhook_endpoints/<endpoint ID>
Header(s)	cookie:dmsess=<cookie-value-returned-from-authentication-api>
Request	

Response	JSON object:
	{
	    "id": <endpoint ID>,
	    "object": "webhook_endpoint",
	    "deleted": true
	}

PeSave APIs

API 1: Retrieve peSave urls

Desc	returns peSave urls for single person or multi person jobs
Method + URI	GET {host}/pesave/getUrls/rid
Header(s)	cookie:dmsess=<cookie-value-returned-from-authentication-api>
Request	

Response	JSON object for a MP job:
	[
	    {
	        "trackingId": "001",
	        "modifiedVersionReadUrl": null,
	        "modifiedVersionWriteUrl": "<URL>"
	    },
	    {
	        "trackingId": "002",
	        "modifiedVersionReadUrl": null,
	        "modifiedVersionWriteUrl": "<URL>"
	    }
	]

Experimental Multi Person API

Multi person feature does not introduce any new api so far. However changes to the existing APIs are mentioned here:

Phase 1: Detect Characters for MP Job

Use the same /**process** restful api with a new parameter.

```
pipeline=mp_detection

{

  “url”: <video upload url>

  “processor”: video2anim

  “params": ["pipeline=mp_detection"]

}
```

No other parameters are required. Existing /**status** api is used to query the status and/or /**download** api to download the characters_detection_result.cdsave and person thumbnails (thumbnail_character_XXX.png) which represent the detected persons in the video and their thumbnails.

The returned rid from this api gets used next for MP job processing, so should be cached/saved in memory at the client side.

Phase 2: Process MP Job

Use the same /**process** restful api with a new argument for a new job (not required for a rerun).
`rid_mp_detection`

and a new json input parameter **models** (a json string) in params object. Include some or all (up to 16) detected persons ids from the characters_detection_result.cdsave as value. The service will run the animation tracking process for those persons. Also associate a 3d model id from the library with each person's detected id to represent the persons as animated 3d models in the output formats.

No need to set the old **model** param which is being used for single person mode only.

```
{

  "rid_mp_detection": <previous MP detection job rid>

  "processor":'video2anim'

  "params": ["models=[{trackingId:'001', modelId:'model_id'}, ...]", ...]

}
```

This is a regular MP job processing sample For a successful job, Existing download and list api response will contain two additional fields:

```
{

  "mode": <0=single person or 1=multi person>

  "models":[{trackingId:<value>, modelId:<value>, faceDataType:<value>,
handDataType:<value>}, ...]

}
```

The returned rid from this api will be the same as the input rid_mp_detection argument.

Rerun will be the same as before. However, for MP jobs, /pesave/getUrls api will return an array containing peSaves for each character.

Download

Downloading the relevant files for MP is similar to Single Person, just a character detection/tracking ID (in the format of _XXX) will be appended for every relevant file name.

Additionally, format wise all character/person files (like output bvh files of all characters are put in a single archive) are also available in the download api response.

```
{
                "name": "all_characters",
                "files": [
                   {
                      "fbx": "<URL>"
                   },
                   {
                      "bvh": "<URL>"
                   },
                {
                    "glb": "<URL>"
                },
                {
                    "dmpe": "<URL>"
                }
            ]
}
```

Animate 3D Restful API Error Codes

Error Code	Meaning
201	Error downloading the video or DM asset
202	Error converting the video
503	Error processing the parameters
504	Error loading the character assets
505	Physics Filter is incompatible with the custom characters
506	Error creating the pose estimation

507	Error while processing the body tracking
508	Input video or image doesn’t meet the requirements to generate animations of good quality
509	Error loading the configurations
510	Error open internal files
511	Processing interrupted
513	Failed to detect character in the video
599	Body tracking timeout
701	Error processing the face tracking
799	Face tracking timeout
901	Error loading the mesh of the custom character
902	Error loading the BVH custom character
903	Error copying animations onto the custom character
904	Error exporting animations for the custom character
905	Custom character doesn’t include skinned mesh information
906	More than half of the required blendshapes are missing
907	Error loading facial definition for the custom character
908	Error loading facial tracking data
909	Error loading the metadata of the custom character

999	Animation baking timeout
1301	Error creating the hand estimation
1302	Error creating the hand estimation
1303	Error creating the hand estimation
1304	Error opening the video
1305	Error parsing video path
1306	Error loading internal files
1307	Error processing hand tracking
1308	Error processing the video
1399	Hand tracking timeout