Saymotion REST API

Revisions

Beta v1.0.0

Initial rest APIs

Beta v1.1.0

/account/creditBalance (introducing feature limits)
/job/process (changes due to variant & inpainting generation)
/job/list (changes due to variant & inpainting generation and also multi mp4 downloads)
/job/download (changes due to variant & inpainting generation)

Beta v1.1.1

/job/process (Added skipFBX parameter)

Beta v1.2.0

/job/process (Added merging parameter)

Beta v1.3.0

/job/import/animate3d (Added new job api to import animate3d jobs to saymotion)

Beta v1.4.0

/job/process (Added support for rerun, refine, loop request)

Beta v1.5.0

/prompt/optimize (Added prompt optimization api)

The SayMotion REST API lets you convert text prompts into 3D animations without having to use the Saymotion Web Portal. It can be used from web, mobile or desktop apps.

Authentication

The SayMotion 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 SayMotion 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 the origin below:
http://localhost

For production deployment, please let us know your production url (scheme, host, port), so that we can configure our CORS setting accordingly.

API Reference

Account APIs

API 1: Authentication & getting 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 expires, this API needs to be called again to get a new session cookie. Please use this api from the server side to stay secure for web applications and you can use the returned session cookie in other API calls even from the client side.
Method + URI	GET {host}/account/v1/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: Credit Balance

Desc	Retrieves Credit Balance for an user
Method + URI	GET {host}/account/v1/creditBalance
Header(s)	cookie:dmsess=<cookie-value-returned-from-authentication-api>
Request	n/a
Response	JSON object: {"credits":<value>,"subscription":{"name":<value>,"credits":<value>,"featureLimits":{"maxVariantsGeneration":<value>},"currentPeriod":{"start":<value>,"end":<value>}}}

Job APIs

API 1: Start Processing

Desc	Start processing text to motion OR rendering a previously generated animation
Method + URI	POST {host}/job/v1/process/processor
Header(s)	cookie:dmsess=<cookie-value-returned-from-authentication-api>

Request	POST body should include a JSON object:
	{
	  “params": [<params>, ...]
	}.

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

Processor Id	Description
text2motion	textPrompt to animation generator
render	Renders an animation to video

<params> specifies additional parameters that will be passed to the specified processor.

For the **text2motion** processor, here are the animation settings parameters for a job.

```
"params":
 [
 ‘model=<value>’,
 ‘dis=<value>’,
 ‘footLockingMode=<value>’,
‘poseFilteringStrength=<value>’,
‘rootAtOrigin=<value>’,
‘skipFBX=<value>’
 ‘numVariant=<value>’ (not applicable for rerun job)
 ]
```

parameters for a regular animation generation job:
 ‘prompt=<value>’

And here are the common parameters for an inpainting , merging, loop, refine & rerun job job

 ‘t2m_rid=<value>’
 ‘variant_id=<value>’

here are the common parameters for an inpainting , merging & loop job
‘editRequest={"numTrimLeft" : <value>, "numTrimRight" : <value>}’

Inpainting exclusive parameter:

 ‘inPaintingRequest={ “prompt” :<value>, “intervals” : [{ “start” :<value>, “end”:<value> }] }’

Merging exclusive parameter::

‘mergingRequest={

```
 “t2m_rid”:<value>,
 “variant_id”:<value>,
“editRequest”:{"numTrimLeft" : <value>, "numTrimRight" : <value>},
“prompt”: <value>,
“blendDuration”: <value>
 }’
```

Refine exclusive parameter::

```
‘refineRequest={
“prompt”: <value> (optional),
“creativity”: <value>
 }’
```

Rerun exclusive parameter::

```
‘rerunRequest={
“rerun”: <0 or 1>
 }’
```

Loop exclusive parameter::

```
‘loopRequest={
“prompt”: <value> (optional),
"numReps": <value>,
“blendDuration”: <value>,
"fixRootMode": <value>,
"fixRootPositionAltitude": <value>,
"fixRootPositionHorizontal": <value>,
"fixRootOrientation": <value>,
"fixAcrossEntireMotion": <value>
 }’
```

prompt
A detailed text prompt to generate motion with. For inpainting, merging & refine jobs, the prompt parameter should be included inside the inPaintingRequest, mergingRequest or refineRequest parameter, instead of as a standalone parameter in a regular generation job.

model
The 3d model used for showcasing the generated motion/animation

dis (optional)
This parameter influences motion generation to improve it in some cases like inter body parts penetration etc, but may cause side

effects in animation quality sometimes. By default simulation is turned on. Use **dis=s** to turn off the simulation.

footLockingMode (optional)

- 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.)

poseFilteringStrength (optional)

- 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 (optional)

- 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

skipFBX (optional)

- Skips FBX format generation. Set skipFBX=1

numVariant (optional)

- How many variants to be generated. Value ranges from 1 to 8. Default is 1 .

t2m_rid

- It is a previous text2motion job request id from which to generate new editing jobs

variant_id

- It is a specific variant animation id from the above previously generated text2motion job

blendDuration : duration in seconds of the merge or loop generated frames

editRequest

- A json string, containing simple edit requests to the inpainting/merging region such as trimming it from left (integer value) or right (integer value). The values represent frame counts and are always positive.

creativity

- Used for refine animation . Range is 0.0 to 1.0. Higher the value, more AI styling will be brought into the original animation.

numReps

- Number of repetition in loop request. value ranges from 1 to any positive number.

fixRootMode

- For loop request. Value should be either 'INTERPOLATION' or 'LOCKED'

fixRootPositionAltitude

- For loop request. Value: 1 (enabled) or 0 (disabled)

fixRootPositionHorizontal

- For loop request. Value: 1 (enabled) or 0 (disabled)

fixRootOrientation

- For loop request. Value: 1 (enabled) or 0 (disabled)

fixAcrossEntireMotion

- For loop request. Value: 1 (entire motion) or 0 (transition area)

inPaintingRequest

- A json string, containing exclusive inpainting related information.

mergingRequest

- A json string, contains exclusive merging related information.

rerunRequest

- A json string, contains exclusive rerun related information

refineRequest

- A json string, contains exclusive refine related information.

loopRequest

- A json string, contains exclusive loop related information.

For the **render** processor, here are the parameters.

Only two required parameters are **t2m_rid** which is a previous text2motion job request id & **variant_id** which is a specific variant animation from the above previously generated text2motion job. Other optional parameters are below:

*To replace the default background with a solid color (for green screening etc.)
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)

*To set a studio like 3d background with a solid color tint
backdrop=studio
bgColor=0,177,64,0

Also, List of supported 2D backdrops:

- Checker_Cloudscape
- Golden_Age_Glitz
- Inferno_Stage
- Jungle_Grove
- Mystic_Brick
- Mystic_Concrete
- Rainbow_Spotlight
- Red_Carpet
- Retro_Revue
- Sakura_Dreamscape
- Spotlit_Stage
- Timbered_Retreat
- Urban_Rooftop

*To enable character shadow
shadow=1 (not applicable for 2D backdrop)

***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

	***camHorizontalAngle** Camera horizontal angle in degrees, where zero means forward camera ● Default value is 0 and range is -90 - +90

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

API 2: Poll for Job Status

<table>
<tr><td><b>Desc</b></td><td>Polls for real-time status of a given processing job</td></tr>
<tr><td><b>Method + URI</b></td><td>GET {host}/job/v1/status/rid
GET {host}/job/v1/status/rid1,rid2,..,rid</td></tr>
<tr><td><b>Header(s)</b></td><td>cookie:dmsess=<cookie-value-returned-from-authentication-api></td></tr>
<tr><td><b>Request</b></td><td>Clients can request the current status of previously submitted processing requests (API3).

Use comma (‘,’) to separate multiple request ids if retrieving status for more than 1 request.</td></tr>
<tr><td><b>Response</b></td><td>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>,
“positionInQueue”: <position in the queue for only PROGRESS status>
}

<status name> is one of the following <b>case sensitive</b> values:
<table><tr><th>Status Name</th><th>Description</th></tr></table></td></tr>
</table>

PROGRESS	Request is still being processed
SUCCESS	Request is processed successfully
FAILURE	Request has failed

<status details> for PROGRESS:

```
{
  “step”: <current step>,
  “total”: <expected total number of steps>
}
```

<status details> for SUCCESS:

```
{
  “In”: <null>,
  “out”: <processed files>
}
```

<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 3: Get Download URLs

Desc	Get download URLs for the specified request ids
Method + URI	GET {host}/job/v1/download/rid?variant_id=1 GET {host}/job/v1//download/rid1,rid2,...,rid?variant_id=2,3,...,n
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. A query parameter called **variant_id** to be able to download specific variant animation related resources. Use comma (‘,’) to separate variant_id values which correspond to individual rid of multiple job 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>,
	  “parameters”: <input params of the job>,
	  “renderJobList” (available only for text2motion jobs):<value>
	  “variantDownloadStatus”: <boolean flag, false when variant id is wrong or user doesn’t has sufficient balance to download this variant data>
	  “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>}
	     ]
	   }
	  ]
	}
	
	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”
	}

API 4: List All Video Processing requests by Status & processor

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}/job/v1/list GET {host}/job/v1/list/status1,...,status/processor
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) along with processor id to retrieve only request ids with the same status value(s). For example, GET /list/PROGRESS/text2motion will only return list of text2motion requests that are still being processed

Response	JSON object:
	{
	"count": <number of records in the rids array>,
	"list": [
	{
	"rid": <job request id>,
	"processingInfo":<detail timing of the processing>,
	"processor":<processor id>,
	"parameters":<job input parameters>,
	“variants” (available only for text2motion jobs): <{“variant_id”:{“download”:<download flag>}}>
	“chargedAmount”: <credits used for this job>
	"status": <status of the job>,
	"ctime": <creation time,milliseconds since epoch>,
	"mtime": <last modification time, milliseconds since epoch>
	},
	... ...
	]
	}

API 5: Cancel progressing job

Desc	Cancel job for the specified request id
Method + URI	GET {host}/job/v1/cancel/rid
Header(s)	cookie:dmsess=<cookie-value-returned-from-authentication-api>
Request	Clients can cancel in progress request/job.

Response	JSON object:
	{
	“result”: true
	}

API 6: Import Animate 3d job to saymotion

Desc	Start importing Animate 3D job. Please use {host}/job/v1/status/rid to get import status.
Method + URI	POST {host}/job//v1/import/animate3d/rid
Header(s)	cookie:dmsess=<cookie-value-returned-from-authentication-api>
Request	POST body should include a JSON object: { model: <saymotion model id> “params": [<params>, ...] (animate3d job params array: modified ones Or if not modified then originals) }.

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

Custom Character APIs

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/v1/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/v1/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 text2motion process API>,
	  “faceDataType”:<if the model supports facial rig>,
	  “handDataType”:<if the model supports hand/finger rig>
	}

API 3: List Models

Desc	List models based on specific query or without
Method + URI	GET {host}/character/v1/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
	  "glb": url of glb format of the character
	  "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/v1/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. Currently only one model can be deleted at a time.
	}

Prompt APIs

API 1: Optimize prompt

<table>
<tr><td><b>Desc</b></td><td>Start processing user prompt to saymotion flavored prompt</td></tr>
<tr><td><b>Method + URI</b></td><td>POST {host}/prompt/v1/optimize</td></tr>
<tr><td><b>Header(s)</b></td><td>cookie:dmsess=<cookie-value-returned-from-authentication-api></td></tr>
<tr><td><b>Request</b></td><td>POST body should include a JSON object:
{
  “params": [<params>, ...]
}.

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

<table>
<tr><th>Processor Id</th><th>Description</th></tr>
<tr><td>llm</td><td>Converts user prompts to saymotion flavored prompt</td></tr>
</table>

<params> specifies additional parameters that will be passed to the specified processor.

For the <b>llm</b> processor, here are the parameters for a regular request.

"params":
 [
 “prompt=<value>”
 “breakIntoActionablePrompts=<1 or 0>” (optional)
 ]</td></tr>
<tr><td><b>Response</b></td><td>JSON object:
{
  "rid": <request id></td></tr>
</table>

	}

API 2: Poll for request status

<table>
<tr><td><b>Desc</b></td><td>Polls for real-time status of a given processing request</td></tr>
<tr><td><b>Method + URI</b></td><td>GET {host}/prompt/v1/status/rid
GET {host}/prompt/v1/status/rid1,rid2,..,rid</td></tr>
<tr><td><b>Header(s)</b></td><td>cookie:dmsess=<cookie-value-returned-from-authentication-api></td></tr>
<tr><td><b>Request</b></td><td>Use comma (‘,’) to separate multiple request ids if retrieving status for more than 1 request.</td></tr>
<tr><td><b>Response</b></td><td>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>,
  “positionInQueue”: <position in the queue for only PROGRESS status>
}

<status name> is one of the following <b>case sensitive</b> values:
<table>
<tr><th>Status Name</th><th>Description</th></tr>
<tr><td>PROGRESS</td><td>Request is still being processed</td></tr>
<tr><td>SUCCESS</td><td>Request is processed successfully</td></tr>
<tr><td>FAILURE</td><td>Request has failed</td></tr>
</table>

<status details> for PROGRESS:</td></tr>
</table>

	{
	“step”: <current step>,
	“total”: <expected total number of steps>
	}
	
	<status details> for SUCCESS:
	{
	“in”: <null>,
	“out”: <saymotion flavored prompt array: a json string>
	}
	
	<status details> for FAILURE include the last error message. Currently the format is:
	{
	“exc_message”: <exception message, if any, error code string array, for example: [“101”]>,
	“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.

Saymotion Restful API Error Codes (Updates as we add more features)

Error Code	Meaning
101	Not enough credit
498	Unknown pipeline error
494	Invalid pipeline Input
501	Error while generating motion

502	Error parsing motion generation parameters
599	Motion generation timeout
603	Error processing pose tracking parameters
604	Error loading animation data for pose tracking
605	Physics Filter is incompatible with used custom character
607	Error while processing the body tracking
610	Error saving pose tracking intermediate results
699	Pose tracking timeout
703	Error processing pose correction parameters
704	Error loading the character animation assets for pose corrections
710	Error saving pose correction intermediate results
799	Pose correction timeout
803	Error processing bvh exporter parameters
804	Error loading the character animation assets for bvh exporting
810	Error saving bvh results
899	Bvh exporting 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
1101	Process stuck
1102	Invalid input parameter
1105	Failed to load input character
1106	Failed to attach animation to character
1107	Failed to configure backdrop