**Automatic Dog Voice Recognition (ADVR) RESTful APIs Specification**

(Draft v0.6)

liesheng.long@gmail.com

**Overview**

This document specifies the API service for Automatic Dog Voice Recognition (ADVR). The initial focus is to prototype matching and tagging of dog audio with its likely intent. Part of this design is purposely simplified. Authentication & Authorization and other necessary elements are left for future. Also, the design is subject to change based on feedback from the team.

**Note**:

1. ${something} means a place holder for something.
2. all relative endpoints in the APIs below are relative to the installation base URL: (http|https)://${server}:${port}/${version},

e.g. https://apis.dogemotiondetector.com

The server (hostname) is different from one env to another. We may choose to support SSL (https) only.

**Versioning**: The version number will evolve overtime. For the demo, it will be v0. In the future, we may support more than one version at the same time in order to support different features and different algorithms. Generally speaking a *newer version* *= older version* + *new features* - *removed features*.

**Authentication**: supports Basic or Bearer token. For the prototype, use Basic authentication. TBD: for real production, will pursue oAuth 2 Authentication. Example of sending Basic auth info can be found here: http://www.avajava.com/tutorials/lessons/how-do-i-connect-to-a-url-using-basic-authentication.html

**Support WAV formats:**

| **WAV format** | **Min** | **Max** |
| --- | --- | --- |
| 32-bit floating-point | -1.0 | +1.0 |
| 32-bit PCM | -2147483648 | +2147483647 |
| 16-bit PCM | -32768 | +32767 |
| 8-bit PCM | 0 | 255 |

**Conventions:**

1. **RESTful**: All APIs are intended to be RESTful.
2. **Encoding**: all data will be utf-8 based when applicable.
3. **Timestamp format**: as Java System.currentTimeMillis(),the difference, measured in milliseconds, between the current time and midnight, January 1, 1970 UTC.
4. **Parameter name**: follows cameral name convention unless otherwise mentioned.

**Parameters**:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | **Type** | **Description** | **Min** | **Max** | **Default** |
| **mediaFile** | **File** | **Video or Wav file. Currently only aforementioned formats are supported.** <1MB, TBD: the max size |  |  |  |
| **mediaFileId** | **String** | **Previously uploaded file ID** |  |  |  |
| **dogId** | **String** | the registered dog Id. it can be any String that contains only alphanumberic and “-“, “\_”. |  |  |  |
| **userId** | String | The registered user Id for the end user |  |  |  |
| **ownerId** | String | the registered owner ID (TBD: be part of the authentication token?) |  |  |  |
| **phoneId** | String | the registered phone ID |  |  |  |
| **emotionHint** | Integer | 1: Normal  2: Funny  3: Annoying |  |  | **1** |
| **recordTime** | long | the time since epoch when the audio was record |  |  | System current time when the API is called |
| **postTime** | long | The time since epoch when the media file is posted or a “like” is posted |  |  | System current time when the API is called |
| **targetLocale** | String | Locale corresponding to the desired translation language: en\_US, zh\_CN, etc. |  |  | en\_US |
| **tag** | String | the tag info, can be in different language. For each language, only one tag is allowed (TBD: more than one?) |  |  |  |
| **tagLocale** | String | Locale corresponding to the desired tag language: en\_US, zh\_CN, etc. |  |  | en\_US |
| **tagTime** | long | the time since epoc when the tag was created for the audio |  |  | System current time when the API is called |
| **likeToggle** | integer | 0--Not Sure  1—Like  2--DisUnlike |  |  |  |

**Input validations**: some of the validation will NOT be implemented for the prototyp**e.**

**APIs**

1. Name: **Upload media for translation**
   1. **Endpoint**: /${version}/cgi-bin/translate
   2. **Description**: this API will call AI library, try to analyze the posted audio file against the database for the closest match. Note: depends on the matching algorithm used, it’s possible there is always a “closest match”.
   3. **Request headers**:
      1. **Authentication**: see “authentication” section above
      2. **Accept**: supports “application/json”
   4. **Request method**: POST
   5. **Parameters**:
      1. **Post Parameters**:
         1. **mediaFile** (required)
         2. **dogId** (required
         3. **userId** (required)
         4. **phoneId** (optional)
         5. **emotionHint** (optional)
         6. **recordTime**(optional)
         7. **targetLocale** (optional)
   6. **Response**: successful response will have a http status code 200 and an “application/json” response body (example):

{

“**mediaFileId**”: “newlyUploadedFileId”, //uuid

“**downloadUrl**”: ”the url for the uploaded media file”,

“**mediaType**”: ”audio|video|image”,

“**mediaFormat**”: ”.wav|.mp4|.gif|.jpg|.jpeg”

“**thumbnail**”: “the url to download the thumbnail for the video” (optional) “**audioDownloadUrl**”: ”the url to down;oad the audio file extracted from the video” (optional)

“**tag**”: “happy”,

“**translation**”: “I am happy”,

“**tagLocale**”: ”en\_US”

}

* 1. **Error codes**:
     1. http status 401: authentication failed
     2. http status 403: the client is not authorized to call this API
     3. http status 404: dogId/ownerId/phoneId isn’t found.
     4. http status 400: some information in the request is bad, e.g., no mediaFile or the file is too big
     5. http status 500: Internal Server Error. Optionally
     6. error response body (example):

{

“**mediaFileId**”: “newlyUploadedFileId”, //uuid, if created successfully

“**errorCode**”: “RECORD\_NOT\_FOUND”,

“**errorMessage**”:”the dog ID is not found”

}

* 1. **Examples**:

POST /v0/translate HTTP/1.1

Host: apis.dogemotiondetector.com:80

Authentication: Bearer some-token

Content-Type: multipart/form-data; boundary=MultipartBoundry

Accept-Encoding: gzip, deflate

-----------------------------229222538423196\r\nContent-Disposition: form-data; name="phoneId"\r\n\r\nphoneId1\r\n-----------------------------229222538423196\r\nContent-Disposition: form-data; name="dogId"\r\n\r\ndogId1\r\n-----------------------------229222538423196\r\nContent-Disposition: form-data; name="ownerId"\r\n\r\nownerId\r\n-----------------------------229222538423196\r\nContent-Disposition: form-data; name="fileToUpload"; filename="sample.wav"\r\nContent-Type: audio/wav\r\n\r\nRIFF¸\n

Example of uploading file can be found here: <http://stackoverflow.com/questions/2469451/upload-files-from-java-client-to-a-http-server>

1. Name: **Translate media by file ID**
   1. **Endpoint**: /${version}/cgi-bin/translate
   2. **Description**: this API will call AI library, try to analyze the pre-uploaded audio file (identified by the mediaFileId) against the database for the closest match. Note: depends on the matching algorithm used, it’s possible there is always a “closest match”.
   3. **Request headers**:
      1. **Authentication**: see “authentication” section above
      2. **Accept**: supports “application/json”
   4. **Request method**: **GET**
   5. **Parameters**:
      1. **Query Parameters**:
         1. **mediaFileId** (required)
         2. **userId** (required)
         3. **phoneId** (optional)
         4. **emotionHint** (optional)
         5. **targetLocale** (optional)
      2. **Post Parameters**: n/a
   6. **Response**: successful response will have a http status code 200 and an “application/json” response body (example):

{

“**mediaFileId**”: “providedByClient”, //uuid

“**tag**”: “I am happy”,

“**translation**”: “I am happy doggy”,

"**emotionHint**": "1",

“**tagtLocale**”:”en\_US”

}

* 1. **Error codes**:
     1. http status 401: authentication failed
     2. http status 403: the client is not authorized to call this API
     3. http status 404: mediaFileId/dogId/ownerId/phoneId isn’t found.
     4. http status 400: some information in the request is bad, e.g., the audio file is too big
     5. http status 500: Internal Server Error. Optionally
     6. error response body (example):

{

“**mediaFileId**”: “providedByClient”, //uuid

“**errorCode**”: “RECORD\_NOT\_FOUND”,

“**errorMessage**”:”the audio file ID is not found”

}

* 1. **Examples**:

Similar to **Translate** API

1. Name: **Tag Media**
   1. **Endpoint**: /${version}/cgi-bin/uploadAndTag
   2. **Description**: this API will tag the posted audio file with known emotion/intent. The backend service will process it and add the results to the database for future matching.
   3. **Request headers**:
      1. **Authentication**: see “authentication” section above
      2. **Accept**: supports “application/json”
   4. **Request method**: POST
   5. **Parameters**:
      1. **Post Parameters**:
         1. **mediaFile** (required)
         2. **dogId** (required)
         3. **tag** (required)
         4. **phoneId** (optional)
         5. **tagTime**(optional)
   6. **Response**: successful response will have a http status code 200 and an “application/json” response body (example):

{

“**mediaFileId**”: “newlyUploadedFileId

“**downloadUrl**”: ”the url for the uploaded media file”,

“**mediaType**”: ”audio|video|image”,

“**mediaFormat**”: ”.wav|.mp4|.gif|.jpg|.jpeg”

“**thumbnail**”: “the url to download the thumbnail for the video” (optional)

“**tagId**”: “newlyCreatedTagId”,

“**status**”: “success

}

* 1. **Error codes**:
     1. http status 401: authentication failed
     2. http status 403: the client is not authorized to call this API
     3. http status 404: dogId/ownerId/phoneId isn’t found
     4. http status 400: some information in the request is bad, e.g., no tags; no mediaFile or the file is too big
     5. http status 500: Internal Server Error. Optionally
     6. error response body (example):

{

“**mediaFileId**”: “newlyUploadedFileId”, //uuid, if created successfully

“**errorCode**”: “INVALID\_INPUT”,

“**errorMessage**”:” Media File is required, Dog ID is required, User ID is required, Tag is required”

}

* 1. **Examples**:

1. Name: **Tag media by media file Id**
   1. **Endpoint**: /${version}/cgi-bin/tag
   2. **Description**: this API will tag the posted audio file with known emotion/intent. The backend service will process it and add the results to the database for future matching.
   3. **Request headers**:
      1. **Authentication**: see “authentication” section above
      2. **Accept**: supports “application/json”
   4. **Request method**: POST
   5. **Parameters**:
      1. **Post Parameters**:
         1. **mediaFileId** (required)
         2. **userId** (required)
         3. **tag** (required)
         4. **phoneId** (optional)
         5. **tagTime**(optional)
   6. **Response**: successful response will have a http status code 200 and an “application/json” response body (example):

{

“**mediaFileId**”: “providedByClient”, //uuid

“**tagId**”: “newlyCreatedTagId”,

“**status**”: “success

}

* 1. **Error codes**:
     1. http status 401: authentication failed
     2. http status 403: the client is not authorized to call this API
     3. http status 404: mediaFileId isn’t found.
     4. http status 400: some information in the request is bad, e.g., no tags; no mediaFile or the file is too big
     5. http status 500: Internal Server Error.
     6. error response body (example):

{

“**mediaFileId**”: “providedByClient”, //uuid, if created successfully

“**errorCode**”: “INVALID\_INPUT”,

“**errorMessage**”:” Media File ID is required; User ID is required; tag ID is required”,

“**status**”: “failed”

“**tagId**”: “newlyCreatedTagId”

}

* 1. **Examples**:

1. Name: **Get Media Info**
   1. **Endpoint**: /${version}/cgi-bin/mediaInfo
   2. **Description**: this API will all the meta data related to the previously uploaded media file, associated tags
   3. **Request headers**:
      1. **Authentication**: see “authentication” section above
      2. **Accept**: supports “application/json”
   4. **Request method**: GET
   5. **Parameters**:
      1. **Query Parameters**:
         1. **mediaFileId** (required)
   6. **Response**: successful response will have a http status code 200 and an “application/json” response body (example):

{

"**mediaFileId**": "2c8d04ff-8fe8-4e10-abd9-9a4dead78410",

"**recordTime**": "1473821665.95",

"**dogId**": "dogId1",

"**mediaType**": "audio",

"**downloadUrl**": "http://10.0.0.126:6080/v0/data/uploaded/2c8d04ff-8fe8-4e10-abd9-9a4dead78410",

"**mediaFormat**": ".wav",

"**rawFile**": "/work/workspace/dog/demo/web/v0/data/uploaded/2c8d04ff-8fe8-4e10-abd9-9a4dead78410",

"**userId**": "123",

"**phoneId**": "",

"**srcFilename**": "dog\_whine.wav",

"**tags**": {

"0": [

{"**tagId**": "a6b7c4ca-eb31-4191-bda2-b7ac65e60233", "**tag**": "happy", "**tagTime**": "1473821888"}],

"**123**": [{"**tagId**": "6f665bdd-4f3c-4ec7-8299-f8fc2eeba709", "tag": "happy", "tagTime": "1473821666"}],

"liesheng": [

{"tagId": "049080a7-d4d3-4ac2-88fd-d5cb9e092919", "tag": "angry", "tagTime": "1474827505", "likes": {}}],

"long": [{"tagId": "43770387-7f81-4d0b-9c15-63abdc0d25b7", "tag": "hungry", "tagTime": "1474827261", "likes":

{"liesheng":

{"postTime": "1474827584", "likeToggle": "2"}}},

{"tagId": "b6d3a79d-4689-4f71-8d4d-923d350f67da", "tag": "bored", "tagTime": "1474827421", "likes":

{"liesheng": {"postTime": "1474827462", "likeToggle": "1"}}}]}

}

}

0: means system tag

* 1. **Error codes:** 
     1. http status 401: authentication failed
     2. http status 403: the client is not authorized to call this API
     3. http status 404: mediaFileId isn’t found.
     4. http status 400: some information in the request is bad, e.g., no tags; no mediaFile or the file is too big
     5. http status 500: Internal Server Error.
     6. error response body (example):

{

"**errorCode**": "INVALID\_INPUT",

"**errorMessage**": "Media File ID is required"

}

* + 1. **Examples**:

1. Name: [Like](http://47.88.106.25/v0/demo/listMedia.html) or Unlike A Tag
   1. **Endpoint**: /${version}/cgi-bin/likeByTagId
   2. **Description**: this API allows user to like or unlike a previous tag
   3. **Request headers**:
      1. **Authentication**: see “authentication” section above
      2. **Accept**: supports “application/json”
   4. **Request method**: GET
   5. **Parameters**:
      1. **Post Parameters**:
         1. **tagId** (required)
         2. **userId** (required)
         3. **phoneId** (optional)
         4. **likeToggle**(required)
   6. **Response**: successful response will have a http status code 200 and an “application/json” response body (example):

{"status": "success", "tagId": "f5800a3d-0b04-407b-a195-5e5f53339bc0"}

* 1. **Error codes:** 
     1. http status 400: some parameters are bad, such as likeToggle
     2. http status 401: authentication failed
     3. http status 403: the client is not authorized to call this API
     4. http status 404: if tagId or userId isn’t found. (not checked for the demo)
     5. http status 500: Internal Server Error.
     6. error response body (example):

{

"**errorCode**": "INVALID\_INPUT",

"**errorMessage**": "Tag ID is required; User ID is required; Please Select An Action: Not Sure/Like/Dislike; Please Select An action: 0--Not Sure/1--Like/2--DisUnlike"

}

* 1. **Examples**:

1. Name: [Get Media List By User Id](http://47.88.106.25/v0/demo/listMedia.html)
   1. **Endpoint**: /${version}/cgi-bin/listMedia
   2. **Description**: this API will list all previously uploaded media files by a user; if user Id isn’t specified, all (be careful since there can be a lot of data)
   3. **Request headers**:
      1. **Authentication**: see “authentication” section above
      2. **Accept**: supports “application/json”
   4. **Request method**: GET
   5. **Parameters**:
      1. **Query Parameters**:
         1. **userId** (optional)
   6. **Response**: successful response will have a http status code 200 and an “application/json” response body (example):

{"2972132": [

{

"downloadUrl": "http://47.88.106.25:80/v0/data/uploaded/b579e2cb-1f74-4a57-b289-5bf5d2d408bb",

"postTime": null,

"mediaType": "video",

"mediaFileId": "b579e2cb-1f74-4a57-b289-5bf5d2d408bb",

"recordTime": ""},

{

"downloadUrl": "http://47.88.106.25:80/v0/data/uploaded/23864658-0882-4d67-bba1-28d3d0085474",

"postTime": null,

"mediaType": "audio",

"mediaFileId": "23864658-0882-4d67-bba1-28d3d0085474", "recordTime": ""},

{

"postTime": "1475027174",

"recordTime": "1475027174",

"mediaType": "video",

"downloadUrl": "http://47.88.106.25:80/v0/data/uploaded/e56642d9-d577-4dfc-bab1-f67b6b8e216a.mp4",

"thumbnail": "http://47.88.106.25:80/v0/data/uploaded/thumbnail/e56642d9-d577-4dfc-bab1-f67b6b8e216a.jpeg",

"mediaFileId": "e56642d9-d577-4dfc-bab1-f67b6b8e216a"

]}

* 1. **Error codes:** 
     1. http status 401: authentication failed
     2. http status 403: the client is not authorized to call this API
     3. http status 500: Internal Server Error.
     4. error response body (not yet specified):
  2. **Examples**: