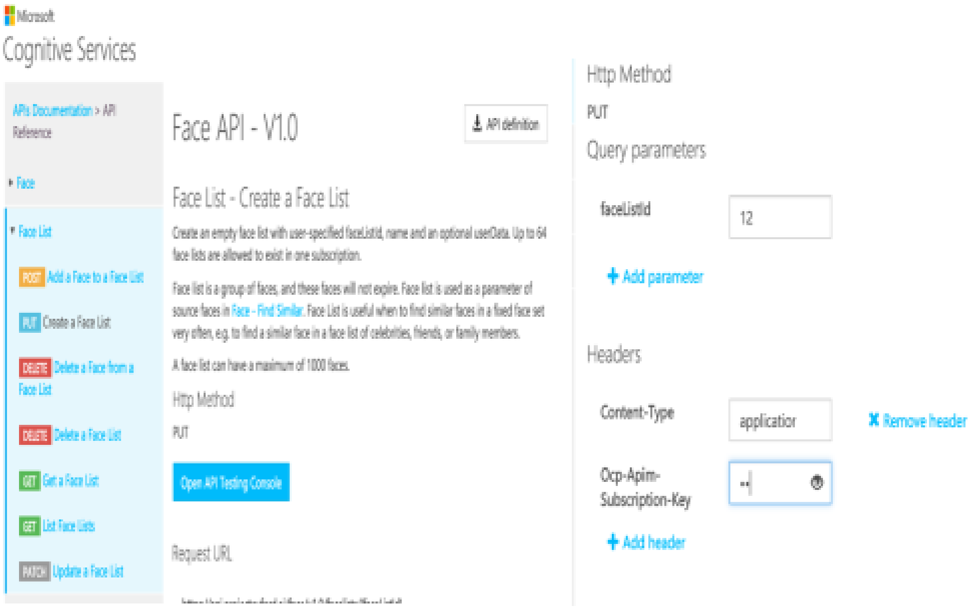
## Technology Demonstrations

### (a) **Microsoft Cognitive Services Face API (REST Service)**



Face API - V1.0

Face List - Create a Face List

Create an empty face list with user-specified faceListId, name and an optional userData. Up to 64 face lists are allowed to exist in one subscription.

Face list is a group of faces, and these faces will not expire. Face list is used as a parameter of source faces in Face - Find Similar. Face List is useful when to find similar faces in a fixed face set very often, e.g. to find a similar face in a face list of celebrities, friends, or family members.

A face list can have a maximum of 1000 faces.

Http Method

PUT

Request URL

Http Method

PUT

Query parameters

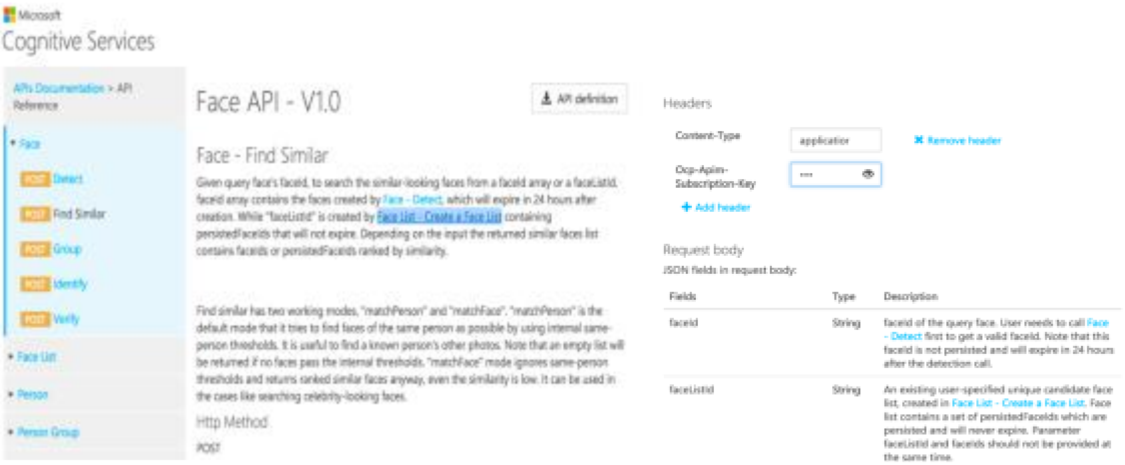
faceListId: 12

Headers

Content-Type: applicatior

Ocp-Apim-Subscription-Key: ••

(To create People Face Database)



Face API - V1.0

Face - Find Similar

Given query face's faceId, to search the similar-looking faces from a faceId array or a faceListId. faceId array contains the faces created by Face - Detect, which will expire in 24 hours after creation. While "faceListId" is created by Face List - Create a Face List containing persistedFaceIds that will not expire. Depending on the input the returned similar faces list contains faceIds or persistedFaceIds ranked by similarity.

Find similar has two working modes, "matchPerson" and "matchFace". "matchPerson" is the default mode that it tries to find faces of the same person as possible by using internal same-person thresholds. It is useful to find a known person's other photos. Note that an empty list will be returned if no faces pass the internal thresholds. "matchFace" mode ignores same-person thresholds and returns ranked similar faces anyway, even the similarity is low. It can be used in the cases like searching celebrity-looking faces.

Http Method

POST

Headers

Content-Type: applicatior

Ocp-Apim-Subscription-Key: •••

Request body

JSON fields in request body:

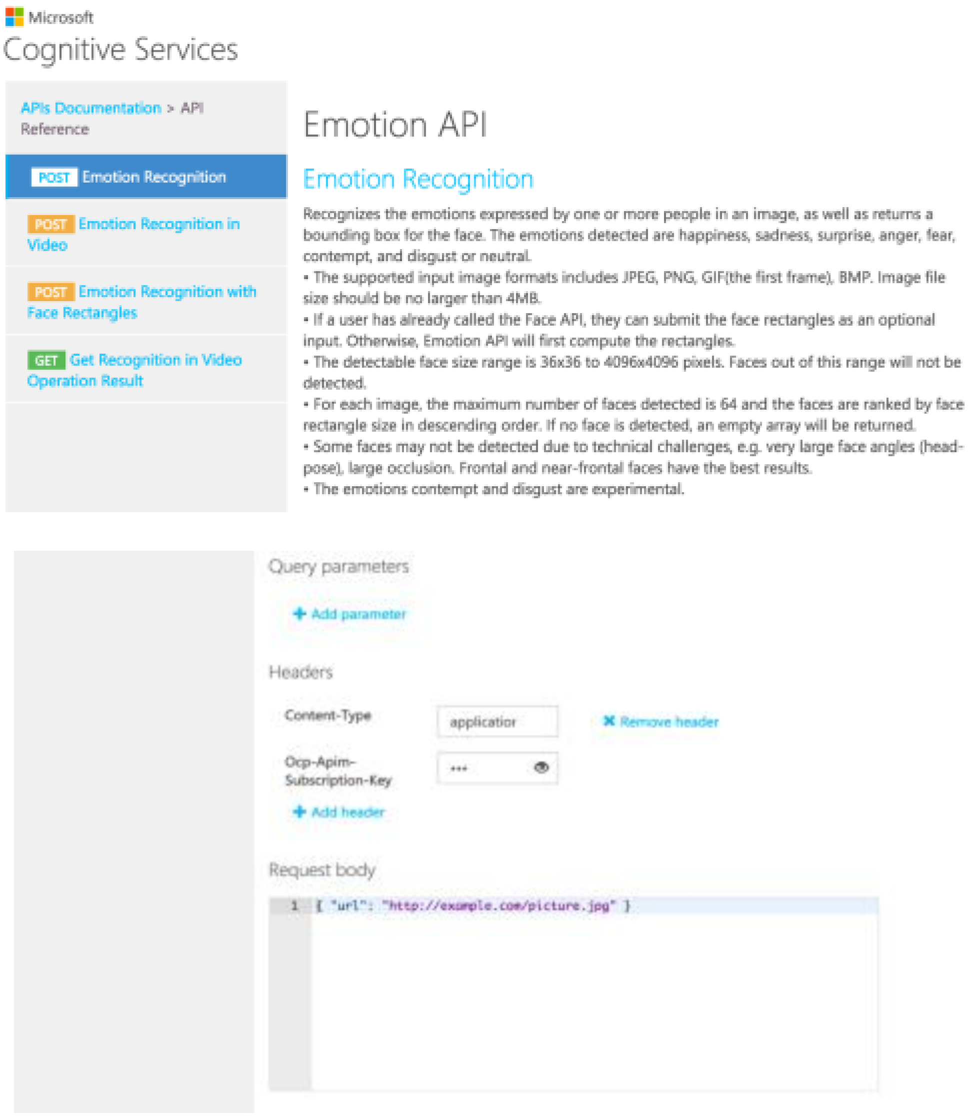
| Fields | Type | Description |
|---|---|---|
| faceId | String | faceId of the query face. User needs to call Face - Detect first to get a valid faceId. Note that this faceId is not persisted and will expire in 24 hours after the detection call. |
| faceListId | String | An existing user-specified unique candidate face list, created in Face List - Create a Face List. Face list contains a set of persistedFaceIds which are persisted and will never expire. Parameter faceListId and faceIds should not be provided at the same time. |

(API for Face Detection and find Similar Face from People Group)

We tested the API online and received correct results as per our project need.

### (b) **Microsoft Cognitive Services Emotion API (REST Service)**

This API is used to recognize the emotion of a face by passing the Base64 string of the image as input parameter and the results is the degree likelihood of possible emotions for that face.



Microsoft
Cognitive Services
APIs Documentation > API Reference
POST Emotion Recognition
POST Emotion Recognition in Video
POST Emotion Recognition with Face Rectangles
GET Get Recognition in Video Operation Result
Emotion API
Emotion Recognition

Recognizes the emotions expressed by one or more people in an image, as well as returns a bounding box for the face. The emotions detected are happiness, sadness, surprise, anger, fear, contempt, and disgust or neutral.

- The supported input image formats includes JPEG, PNG, GIF(the first frame), BMP. Image file size should be no larger than 4MB.
- If a user has already called the Face API, they can submit the face rectangles as an optional input. Otherwise, Emotion API will first compute the rectangles.
- The detectable face size range is 36x36 to 4096x4096 pixels. Faces out of this range will not be detected.
- For each image, the maximum number of faces detected is 64 and the faces are ranked by face rectangle size in descending order. If no face is detected, an empty array will be returned.
- Some faces may not be detected due to technical challenges, e.g. very large face angles (head-pose), large occlusion. Frontal and near-frontal faces have the best results.
- The emotions contempt and disgust are experimental.

Content-Type: applicatior

Ocp-Apim-Subscription-Key: •••

```
{ "url": "http://example.com/picture.jpg" }
```

(API for Emotion Detection)

(c) **Tessel Camera Module (JavaScript Library)**

NPM Module: **npm install camera-vc0706**

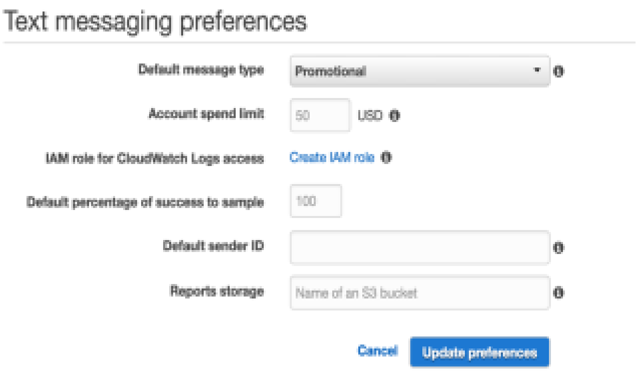
Reference: http://start.tessel.io/modules/camera

We have tested this module on our tessel hardware and were able to communicate with this device after installing this device and calling the module in JavaScript file.

(d) **Web App (Dashboard View) – Python/JavaScript/HTML**

Using JavaScript or Python, develop a website (hosted webpage to display our results as well as trigger some user actions – using HTML and JavaScript controls).

(e) **Sending SMS (Amazon SNS)**



Text messaging preferences

Default message type: Promotional

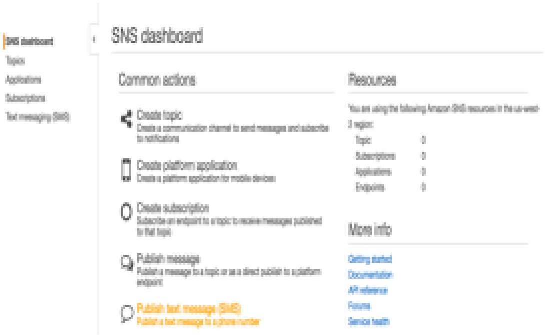
Account spend limit: 50 USD

IAM role for CloudWatch Logs access: Create IAM role

Default percentage of success to sample: 100

Default sender ID:

Reports storage: Name of an S3 bucket



SNS dashboard

Common actions

Create topic
Create a communication channel to send messages and subscribe to notifications

Create platform application
Create a platform application for mobile devices

Create subscription
Subscribe an endpoint to a topic to receive messages published to that topic

Publish message
Publish a message to a topic or as a direct publish to a platform endpoint

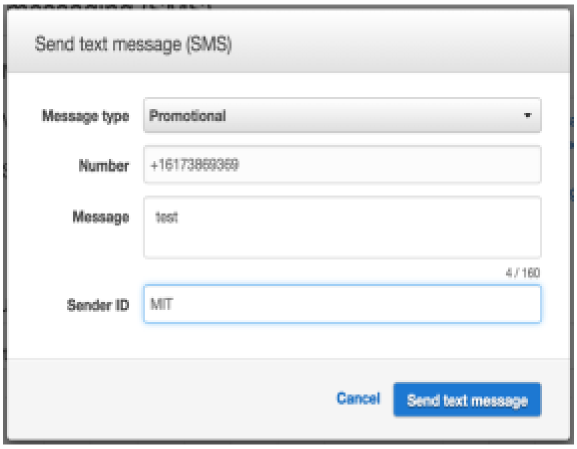
Publish text message (SMS)
Publish a text message to a phone number

Resources

You are using the following Amazon SNS resources in the us-west-2 region:

| | |
|---|---|
| Topic | 0 |
| Subscriptions | 0 |
| Applications | 0 |
| Endpoints | 0 |

More info



Send text message (SMS)

Message type: Promotional

Number: +16173869369

Message: test

4 / 160

Sender ID: MIT

(f) **Web Server (Django or Node.js)**

Installation of a web framework on an Amazon EC2 instance.

(g) **DBMS (MySQL Server)**

Define Table Schema and create the tables in RDMS (MySQL) database in AWS.

(h) **AWS Analytics API**

Tested the API online by passing some input parameters and analyzing results. Based on our progress, we will design appropriate prediction model and then use this service in building the model.

(i) **Infrastructure (AWS)**

Account creation and set-up the environment. Also, choose the services we want for our project (one-year free trial service).