**SinglePlatform Matching API Documentation**

**Introduction**

The SinglePlatform Matching API enables you to match your own list of businesses against locations in the SinglePlatform database. It takes the list of your business locations, performs the matching according to the matching criteria you specify, and returns the matching results along with other optional information.

Because the API uses the address normalization API from SmartyStreets (<http://smartystreets.com>), matching is currently limited to US-based businesses.

\*Please note that this Matching API is currently in beta. In the coming weeks, SinglePlatform will be adding additional functionality. We highly encourage feedback from our partners; please do not hesitate to reach out with suggestions or issues.

**Partner Agreement Considerations**

Our partners are required to match and add SinglePlatform’s new locations on the 1st and 15th of every month, at a minimum. SinglePlatform has built this matching service to help our partners increase their matching rates against SinglePlatform’s locations dataset.

Matt Doumar (<mdoumar@singleplatform.com>), SinglePlatform’s product lead, will be available to work with our partners' dev teams to ensure the process of matching / updating / adding locations is as straightforward as possible and in compliance with SinglePlatform's partner agreement.

**Authentication**

A valid client ID is required as a URL parameter (client=<your\_client\_id>) to authenticate and authorize you to use the API.

**The Request**

URL: <http://matching-api.singleplatform.com/location-match>

The Matching API server expects HTTP POST requests and no data should be specified in the URL.

Each POST request should include a JSON-formatted request body with the following keys:

**matching\_criteria**: String, optional [default=’NAME\_ADDRESS’]

*Determines what criteria to use when matching your businesses against the SinglePlatform database of locations.*

* Each location is expected to include the key elements according to the chosen matching criteria.
* There are five options for matching criteria:
  + NAME\_ADDRESS:
    - Fuzzy matching is run on a combination of the ‘Name’ and ‘Address’ fields
    - Required fields: 'name', 'address'
    - The following combinations are considered valid addresses:
      * Street address + City + State + Zip
      * Street address + Zip
      * Street address + City + State
    - Format requirements: The address fields can be separated by spaces or commas
  + NAME\_GEOLOCATION:
    - Fuzzy matching is run on a combination of the ‘Name’ and ‘Geolocation’ fields (i.e., Latitude & Longitude values)
    - Required fields: 'name', 'latitude', 'longitude'
  + NAME\_PHONE:
    - An exact match must be found on the ‘phone number’ value and then a fuzzy match is performed on the ‘Name’ value.
    - Required fields: 'name', 'phone'
    - Format requirements: The phone number must be 10 digits. If any additional characters are included (e.g., parentheses), they will be stripped
  + FOURSQUARE\_ID:
    - An exact match must be found on the ‘foursquare\_id’ value.
    - Note that SinglePlatform does not currently have complete coverage of valid Foursquare IDs currently. If you do not receive a match on Foursquare ID, we suggest matching again with different criteria.
    - Required field: 'foursquare\_id’
  + FACTUAL\_ID:
    - An exact match must be found on the ‘factual\_id’ value.
    - Note that SinglePlatform does not currently have complete coverage of valid Foursquare IDs. If you do not receive a match on Foursquare ID, we suggest matching again with different criteria.
    - Required field: ‘factual\_id’
* You can specify multiple matching criteria for your businesses, e.g. “matching\_criteria= NAME\_ADDRESS,FOURSQUARE\_ID”. The API will try to match your businesses using the first criterion you specify, then do a second round of matching using the second criterion you specify for the businesses that cannot be matched, and so on and so forth.

**locations**: list of JSON arrays, *required*

*This is the list of locations that you wish to match.*

* Malformed locations will be skipped.
* If matching by “NAME\_ADDRESS” (see “**matching\_criteria**”), the number of locations cannot exceed 100 for each request, and locations with addresses that cannot be normalized will be skipped from matching.

**multiple\_matches**: Boolean, *optional* [default=False]

*Determines if returning multiple matches for one location.*

* If True, up to 5 matches are returned for each location, and will be returned in order of confidence in the match.
* If False, will return the best match above the cutoff\_threshold (see below).

**cutoff\_threshold**: Integer [0,100], *optional* [default=75]

*Determines the accuracy threshold of the fuzzy matching.*

* When a single best match is returned, it will include a key called *match\_accuracy*, which specifies the accuracy of the match.

**geo\_radius**: Float, *optional* [default=0.001]

*Determines the radius around the given geo location pair (lat/long). To be used with the NAME\_GEOLOCATION matching option.*

**The Response**

After the API receives a properly formatted request, it attempts to match every location. The response is a JSON object with a single key “response”.

The response key will contain either a list of matching results or error information (error\_code and error\_message).

If there is no error and a list of matching results is returned, each result will contain the ‘spv2id’, which is the unique location ID used in SinglePlatform’s database. If one single match is found, the value will be the corresponding ID; if multiple matches are found, the value will be the IDs (comma-separated) of the matches, in descending order of the accuracy; if no match is found, the value will be *False*.

**Examples**

Send HTTP POST requests to URL [http://matching-api.singleplatform.com/location-match?client=<your\_client\_id](http://matching-api.singleplatform.com/location-match?client=%3cyour_client_id)> with the following request bodies.

1. Matching by **NAME\_ADDRESS**:

{

"locations":[

{

"address":"2047 US Highway 45 Byp S Trenton TN 38382",

"name":"Dairy Queen"

},

{

"address":"1125 W Riverdale Road Riverdale Utah",

"name":"Applebees Neighborhood Grill"

},

{

"address":"15479 Hwy. One Marshall California",

"name":"Tomales Bay Oyster Company"

},

{

"address":"1825 Washington Rd, Washington PA 15301",

"name":"Primo"

}

],

"matching\_criteria":"NAME\_ADDRESS"

}

Response:

{

"response": [

{

"normalized\_address": "2047 US Highway 45 Byp S Trenton TN 38382",

"spv2id": "dairy-queen-4584",

"match\_accuracy": 100,

"name": "Dairy Queen",

"address": "2047 US Highway 45 Byp S Trenton TN 38382"

},

{

"normalized\_address": "1125 W Riverdale Rd Riverdale UT 84405",

"spv2id": "applebees-1846",

"match\_accuracy": 81,

"name": "Applebees Neighborhood Grill",

"address": "1125 W Riverdale Road Riverdale Utah"

},

{

"spv2id": false,

"name": "Tomales Bay Oyster Company",

"address": "15479 Hwy. One Marshall California"

},

{

"normalized\_address": "1825 Washington Rd Washington PA 15301",

"spv2id": "primo-restaurant",

"match\_accuracy": 100,

"name": "Primo",

"address": "1825 Washington Rd, Washington PA 15301"

}

]

}

2. Matching by **NAME\_GEOLOCATION**:

{

"locations":[

{

"latitude":55.061160000000001,

"longitude":-162.30706000000001,

"name":"King Cove Chinese Restaurant"

},

{

"latitude":21.934650000000001,

"longitude":-159.52654999999999,

"name":"Pomodoro"

}

],

"matching\_criteria":"NAME\_GEOLOCATION",

"geo\_radius":"0.01",

}

Response:

{

"response": [

{

"latitude": 55.06116,

"spv2id": false,

"name": "King Cove Chinese Restaurant",

"longitude": -162.30706

},

{

"latitude": 21.93465,

"spv2id": "pomodoro-39",

"match\_accuracy": 100,

"name": "Pomodoro",

"longitude": -159.52655

}

]

}

3. Matching by **NAME\_PHONE**:

{

"matching\_criteria":"NAME\_PHONE",

"locations":[

{

"name":"Chen Test Restaurant",

"phone":"212-215-0212"

},

{

"name":"Pomodoro",

"phone":"561.496.3503"

}

]

}

Response:

{

"response": [

{

"spv2id": false,

"phone": "212-215-0212",

"name": "Chen Test Restaurant"

},

{

"spv2id": "pomodoro-72",

"phone": "561.496.3503",

"match\_accuracy": 100,

"name": "Pomodoro"

}

]

}

4. Matching by **FOURSQUARE\_ID**:

{

"matching\_criteria":"FOURSQUARE\_ID",

"locations":[

{

"foursquare\_id":"3fd66200f964a52000ea1ee3"

},

{

"name":"Pomodoro",

"foursquare\_id":"3fd66200f964a52000ec1ee3"

}

]

}

Response:

{

"response": [

{

"spv2id": "images-219",

"foursquare\_id": "3fd66200f964a52000ea1ee3"

},

{

"spv2id": false,

"foursquare\_id": "3fd66200f964a52000ec1ee3"

}

]

}

5. Matching by **FACTUAL\_ID**:

{

"matching\_criteria":"FACTUAL\_ID",

"locations":[

{

"factual\_id":"c68bcf4b-c0ba-4c13-b91f-13dd3f26a03b"

},

{

"name":"test",

"factual\_id":"53a5a2ec-bf1e-4e78-88ab-c7cc254faa66"

},

{

"factual\_id":"53a5abcd-abcd-4e78-88ab-c7cc254faa45"

}

]

}

Response:

{

"response": [

{

"spv2id": "ottos-cafe-at-ottos-cafe",

"factual\_id": "c68bcf4b-c0ba-4c13-b91f-13dd3f26a03b"

},

{

"spv2id": "dairy-queen-4584",

"factual\_id": "53a5a2ec-bf1e-4e78-88ab-c7cc254faa66"

},

{

"spv2id": false,

"factual\_id": "53a5abcd-abcd-4e78-88ab-c7cc254faa45"

}

]

}

6. Matching by **NAME\_ADDRESS & NAME\_PHONE**:

{

"matching\_criteria":"NAME\_ADDRESS,NAME\_PHONE",

"locations":[

{

"name":"Dairy Queen",

"address":"2047 US Highway 45 Byp S Trenton TN 38382"

},

{

"name":"Primo",

"address":"1825 Washington Rd, Washington PA 15301"

},

{

"name":"chenstest",

"address":"17 battery pl"

},

{

"name":"Pomodoro",

"phone":"561.496.3503"

}

]

}

Response:

{

"response": [

{

"normalized\_address": "2047 US Highway 45 Byp S Trenton TN 38382",

"spv2id": "dairy-queen-4584",

"match\_accuracy": 100,

"name": "Dairy Queen",

"address": "2047 US Highway 45 Byp S Trenton TN 38382"

},

{

"normalized\_address": "1825 Washington Rd Washington PA 15301",

"spv2id": "primo-restaurant",

"match\_accuracy": 100,

"name": "Primo",

"address": "1825 Washington Rd, Washington PA 15301"

},

{

"spv2id": false,

"name": "chenstest",

"address": "17 battery pl"

},

{

"spv2id": "pomodoro-72",

"phone": "561.496.3503",

"match\_accuracy": 100,

"name": "Pomodoro"

}

]

}

7. Matching by **NAME\_GEOLOCATION** & **FOURSQUARE\_ID**:

{

"locations":[

{

"foursquare\_id":"3fd66200f964a52000ea1ee3"

},

{

"foursquare\_id":"3fd66200f964a52000ec1ee3",

"name":"Pomodoro"

},

{

"latitude":55.061160000000001,

"longitude":-162.30706000000001,

"name":"King Cove Chinese Restaurant"

},

{

"latitude":21.934650000000001,

"longitude":-159.52654999999999,

"name":"Pomodoro"

}

],

"matching\_criteria":"NAME\_GEOLOCATION,FOURSQUARE\_ID"

}

Response:

{

"response":[

{

"latitude":55.061160000000001,

"longitude":-162.30706000000001,

"name":"King Cove Chinese Restaurant",

"spv2id":false

},

{

"latitude":21.934650000000001,

"longitude":-159.52654999999999,

"match\_accuracy":100,

"name":"Pomodoro",

"spv2id":"pomodoro-39"

},

{

"foursquare\_id":"3fd66200f964a52000ea1ee3",

"spv2id":"images-219"

},

{

"foursquare\_id":"3fd66200f964a52000ec1ee3",

"spv2id":false

}

]

}

8. Mal-formatted JSON Requests:

{

"locations"**=**[

{

"foursquare\_id":"3fd66200f964a52000ea1ee3"

}

],

"matching\_criteria":"FOURSQUARE\_ID"

}

Response:

{

"response": {

"error\_message": "Mal-formatted JSON request body.",

"error\_code": "INVALID\_JSON"

}

}