Table of Contents

Searching the web for entities and places

3/5/2018 • 9 min to read • Edit Online

The Entity Search API sends a search query to Bing and gets results that include entities and places. Place results include restaurants, hotel, or other local businesses. For places, the query can specify the name of the local business or it can ask for a list (for example, restaurants near me). Entity results include persons, places, or things. Place in this context is tourist attractions, states, countries, etc.

Search query term

If you provide a search box where the user enters their search term, use the Bing Autosuggest API to improve the experience. The API returns suggested query strings based on partial search terms as the user types.

After the user enters their query term, URL encode the term before setting the q query parameter. For example, if the user enters *Bill Gates*, set `q` to *Bill+Gates* or *Bill%20Gates*.

If the query term contains a spelling mistake, the search response includes a QueryContext object. The object shows the original spelling and the corrected spelling that Bing used for the search.

```
"queryContext": {
  "originalQuery": "Bill Gares",
  "alteredQuery": "bill gates",
  "alterationOverrideQuery": "+Bill Gares",
  "adultIntent": false
}
```

You can use this information to let the user know that you modified their query string when you display the search results.

Requesting entities

For an example request, see Making your first request.

The response contains a SearchResponse object. If Bing finds an entity or place that's relevant, the object includes the `entities` field, `places` field, or both. If Bing does not find relevant entities, the response object will not include the fields.

The `entities` field is an EntityAnswer object that contains a list of Entity objects (see the `value` field). The list may contain a single dominant entity, multiple disambiguation entities or both. A dominant entity is an entity that Bing believes is the only entity that satisfies the request (there is no ambiguity as to which entity satisfies the request). If multiple entities could satisfy the request, the list will contain more than one disambiguation entity. For example, if the request uses the generic title of a movie franchise, the list would likely contain disambiguation entities. But, if the request specifies a specific title from the franchise, the list would likely contain a siingle dominant entity.

Entities include persons such as Adam Levine, places such as Mount Rainier or Pike Place Market, and things such as banana, goldendoodle, book, or movie title. The entityPresentationInfo field contains hints that identify the entity's type. For example, if it's a person, movie, animal, or attraction. For a list of possible types, see Entity Types

```
            "entityPresentationInfo" : {
                "entityScenario" : "DominantEntity",
                "entityTypeHints" : ["Attraction"],
                "entityTypeDisplayHint" : "Mountain"
            },
```

The following shows a response that includes a dominant and disambiguation entities.

```
{
    "_type" : "SearchResponse",
    "queryContext" : {
        "originalQuery" : "pike place market"
    },
    "entities" : {
        "value" : [{
            "contractualRules" : [{
                "_type" : "ContractualRules\/LicenseAttribution",
                "targetPropertyName" : "description",
                "mustBeCloseToContent" : true,
                "license" : {
                    "name" : "CC-BY-SA",
                    "url" : "http:\/\/creativecommons.org\/licenses\/by-sa\/3.0\/"
                },
                "licenseNotice" : "Text under CC-BY-SA license"
            },
            {
                "_type" : "ContractualRules\/LinkAttribution",
                "targetPropertyName" : "description",
                "mustBeCloseToContent" : true,
                "text" : "en.wikipedia.org",
                "url" : "http:\/\/en.wikipedia.org\/wiki\/Pike_Place_Market"
            },
            {
                "_type" : "ContractualRules\/MediaAttribution",
                "targetPropertyName" : "image",
                "mustBeCloseToContent" : true,
                "url" : "http:\/\/en.wikipedia.org\/wiki\/Pike_Place_Market"
            }],
            "webSearchUrl" : "https:\/\/www.bing.com\/search?q=Pike%20Place%20Market...",
            "name" : "Pike Place Market",
            "url" : "http:\/\/www.pikeplacemarket.org\/",
            "image" : {
                "name" : "Pike Place Market",
                "thumbnailUrl" : "https:\/\/www.bing.com\/th?id=A4ae343983daa4...",
                "provider" : [{
                    "_type" : "Organization",
                    "url" : "http:\/\/en.wikipedia.org\/wiki\/Pike_Place_Market"
                }],
                "hostPageUrl" : "http:\/\/upload.wikimedia.org\/wikipedia\/commons\/7\/72\/Pike_Place...",
                "width" : 110,
                "height" : 110
            },
            "description" : "Pike Place Market is a public market overlooking the Elliott...",
            "entityPresentationInfo" : {
                "entityScenario" : "DominantEntity",
                "entityTypeHints" : ["Attraction"]
            },
            "bingId" : "38b9431e-cf91-93be-0584-c42a3ecbfdc7"
        },
        {
            "contractualRules" : [{
                "_type" : "ContractualRules\/MediaAttribution",
                "targetPropertyName" : "image",
                "mustBeCloseToContent" : true,
                "url" : "http:\/\/en.wikipedia.org\/wiki\/Pike_Place_Fish_Market"
            }],
            "webSearchUrl" : "https:\/\/www.bing.com\/search?q=Pike%20Place%20Fish%20Market...",
            "name" : "Pike Place Fish Market",
            "url" : "http:\/\/pikeplacefish.com\/",
            "image" : {
                "name" : "Pike Place Fish Market",
                "thumbnailUrl" : "https:\/\/www.bing.com\/th?id=A91bdc5a1b648a695a39...",
                "provider" : [{
                    "_type" : "Organization",
                    "url" : "http:\/\/en.wikipedia.org\/wiki\/Pike_Place_Fish_Market"
                }],
                "hostPageUrl" : "http:\/\/upload.wikimedia.org\/wikipedia\/en\/7\/7a...",
                "width" : 50,
                "height" : 50
            },
            "description" : "The Pike Place Fish Market, founded in 1930, is an open air fish market...",
            "entityPresentationInfo" : {
                "entityScenario" : "DisambiguationItem",
                "entityTypeHints" : ["Organization"]
            },
            "bingId" : "29d4b681-227a-3924-7bb1-8a54e8666b8c"
        }]
    }
}
```

The entity includes a `name`, `description`, and `image` field. When you display these fields in your user experience, you must attribute them. The `contractualRules` field contains a list of attributions that you must apply. The contractual rule identifies the field that the attribution applies to. For information about applying attribution, see Attribution.

```
            "contractualRules" : [{
                "_type" : "ContractualRules\/LicenseAttribution",
                "targetPropertyName" : "description",
                "mustBeCloseToContent" : true,
                "license" : {
                    "name" : "CC-BY-SA",
                    "url" : "http:\/\/creativecommons.org\/licenses\/by-sa\/3.0\/"
                },
                "licenseNotice" : "Text under CC-BY-SA license"
            },
            {
                "_type" : "ContractualRules\/LinkAttribution",
                "targetPropertyName" : "description",
                "mustBeCloseToContent" : true,
                "text" : "en.wikipedia.org",
                "url" : "http:\/\/en.wikipedia.org\/wiki\/Mount_Rainier"
            },
            {
                "_type" : "ContractualRules\/MediaAttribution",
                "targetPropertyName" : "image",
                "mustBeCloseToContent" : true,
                "url" : "http:\/\/en.wikipedia.org\/wiki\/Mount_Rainier"
            }],
```

When you display the entity information (name, description, and image), you must also use the URL in the `webSearchUrl` field to link to the Bing search results page that contains the entity.

The `places` field is a LocalEntityAnswer object that contains a list of Place objects (see the `value` field). The list contains one or more local entities that satisfy the request.

Places include restaurant, hotels, or local businesses. The entityPresentationInfo field contains hints that identify the local entity's type. The list contains a list of hints such as Place, LocalBusiness, Restaurant. Each successive hint in the array narrows the entity's type. For a list of possible types, see Entity Types

```
            "entityPresentationInfo" : {
                "entityScenario" : "ListItem",
                "entityTypeHints" : ["Place",
                "LocalBusiness",
                "Restaurant"]
            },
```

Local aware entity queries such as *restaurant near me* require the user's location to provide accurate results. Your requests should always use the X-Search-Location and X-MSEdge-ClientIP headers to specify the user's location. If Bing thinks the query would benefit from the user's location, it sets `askUserForLocation` field of QueryContext to **true**.

```
{
    "_type" : "SearchResponse",
    "queryContext" : {
        "originalQuery" : "cafe flora",
        "askUserForLocation" : true
    },
    . . .
}
```

A place result includes the place's name, address, telephone number, and URL to the entity's website. When you display the entity information, you must also use the URL in the `webSearchUrl` field to link to the Bing search results page that contains the entity.

```
    "places" : {
        "value" : [{
            "_type" : "Restaurant",
            "webSearchUrl" : "https:\/\/www.bing.com\/search?q=Cafe%20Flora...",
            "name" : "Cafe Flora",
            "url" : "http:\/\/cafeflora.com\/",
            "entityPresentationInfo" : {
                "entityScenario" : "ListItem",
                "entityTypeHints" : ["Place",
                "LocalBusiness",
                "Restaurant"]
            },
            "address" : {
                "addressLocality" : "Seattle",
                "addressRegion" : "WA",
                "postalCode" : "98112",
                "addressCountry" : "US",
                "neighborhood" : "Madison Park"
            },
            "telephone" : "(206) 325-9100"
        }]
```

NOTE

You, or a third party on your behalf, may not use, retain, store, cache, share, or distribute any data from the Entities API for the purpose of testing, developing, training, distributing or making available any non-Microsoft service or feature.

Throttling requests

The service and your subscription type determines the number of queries that you may make per second (QPS). You should ensure that your application includes the logic necessary to stay within your quota. If you exceed your QPS, the request fails with HTTP status code 429. The response also includes the Retry-After header, which contains the number of seconds that you should wait before sending another request.

Denial of Service (DOS) versus Throttling

The service differentiates between a DOS attack and QPS violation. If the service suspects a denial of service attack, the request succeeds (HTTP status code is 200 OK); however, the body of the response is empty.

Data attribution

Bing Entity API responses contain information owned by third parties. You are responsible to ensure your use is appropriate, for example by complying with any creative commons license your user experience may rely on.

If an answer or result includes the `contractualRules`, `attributions`, or `provider` fields, you must attribute the data. If the answer does not include any of these fields, no attribution is required. If the answer includes the `contractualRules` field and the `attributions` and/or `provider` fields, you must use the contractual rules to attribute the data.

The following example shows an entity that includes a MediaAttribution contractual rule and an Image that includes a `provider` field. The MediaAttribution rule identifies the image as the target of the rule, so you'd ignore the image's `provider` field and instead use the MediaAttribution rule to provide attribution.

```
        "value" : [{
            "contractualRules" : [
                . . .
                {
                    "_type" : "ContractualRules\/MediaAttribution",
                    "targetPropertyName" : "image",
                    "mustBeCloseToContent" : true,
                    "url" : "http:\/\/en.wikipedia.org\/wiki\/Space_Needle"
                }
            ],
            . . .
            "image" : {
                "name" : "Space Needle",
                "thumbnailUrl" : "https:\/\/www.bing.com\/th?id=A46378861201...",
                "provider" : [{
                    "_type" : "Organization",
                    "url" : "http:\/\/en.wikipedia.org\/wiki\/Space_Needle"
                }],
                "hostPageUrl" : "http:\/\/www.citydictionary.com\/Uploaded...",
                "width" : 110,
                "height" : 110
            },
            . . .
        }]
```

If a contractual rule includes the `targetPropertyName` field, the rule applies only to the targeted field. Otherwise, the rule applies to the parent object that contains the `contractualRules` field.

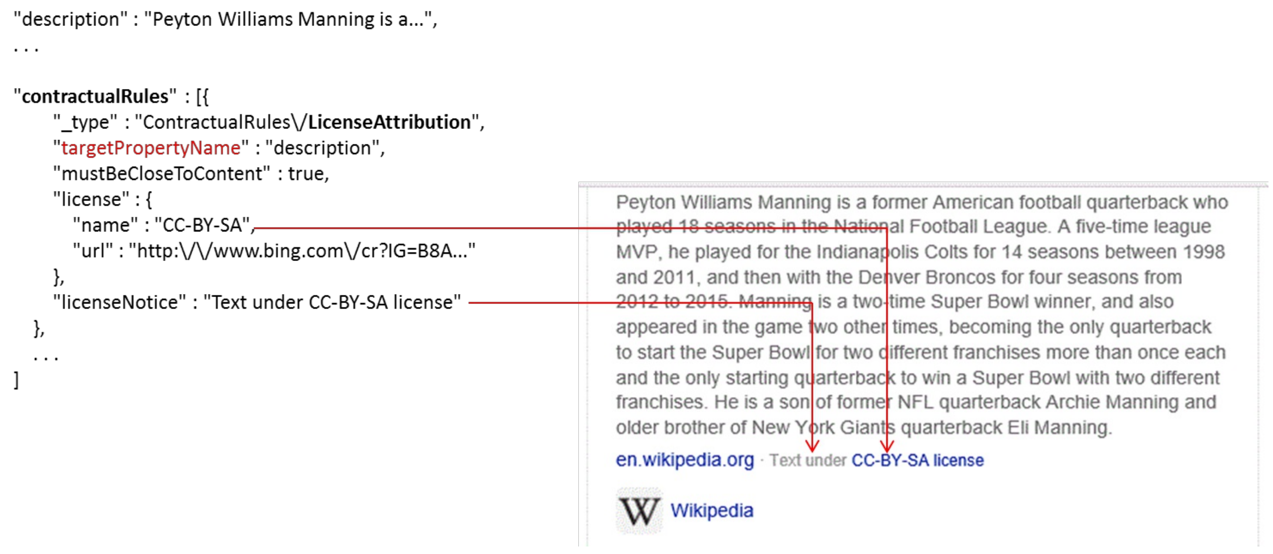
In the following example, the `LinkAttribution` rule includes the `targetPropertyName` field, so the rule applies to the `description` field. For rules that apply to specific fields, you must include a line immediately following the targeted data that contains a hyperlink to the provider's website. For example, to attribute the description, include a line immediately following the description text that contains a hyperlink to the data on the provider's website, in this case create a link to en.wikipedia.org.

```
"entities" : {
    "value" : [{
            . . .
            "description" : "Peyton Williams Manning is a former American....",
            . . .
            "contractualRules" : [{
                    "_type" : "ContractualRules\/LinkAttribution",
                    "targetPropertyName" : "description",
                    "mustBeCloseToContent" : true,
                    "text" : "en.wikipedia.org",
                    "url" : "http:\/\/www.bing.com\/cr?IG=B8AD73..."
                },
            . . .
```

License Attribution

If the list of contractual rules includes a LicenseAttribution rule, you must display the notice on the line immediately following the content that the license applies to. The `LicenseAttribution` rule uses the `targetPropertyName` field to identify the property that the license applies to.

The following shows an example that includes a `LicenseAttribution` rule.



```
"description" : "Peyton Williams Manning is a...",
. . .

"contractualRules" : [{
    "_type" : "ContractualRules\/LicenseAttribution",
    "targetPropertyName" : "description",
    "mustBeCloseToContent" : true,
    "license" : {
      "name" : "CC-BY-SA",
      "url" : "http:\/\/www.bing.com\/cr?IG=B8A..."
    },
    "licenseNotice" : "Text under CC-BY-SA license"
  },
  . . .
]
```

Peyton Williams Manning is a former American football quarterback who played 18 seasons in the National Football League. A five-time league MVP, he played for the Indianapolis Colts for 14 seasons between 1998 and 2011, and then with the Denver Broncos for four seasons from 2012 to 2015. Manning is a two-time Super Bowl winner, and also appeared in the game two other times, becoming the only quarterback to start the Super Bowl for two different franchises more than once each and the only starting quarterback to win a Super Bowl with two different franchises. He is a son of former NFL quarterback Archie Manning and older brother of New York Giants quarterback Eli Manning.

en.wikipedia.org · Text under CC-BY-SA license

Wikipedia

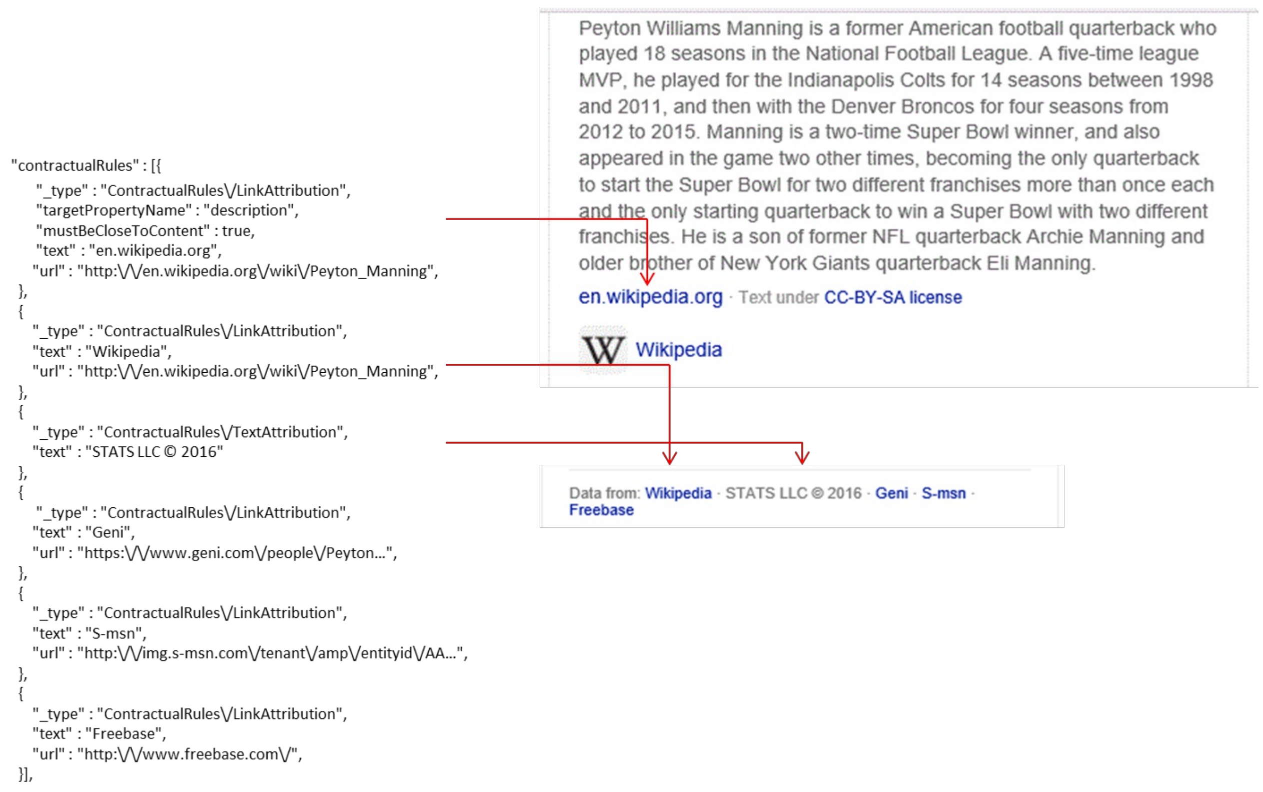
The license notice that you display must include a hyperlink to the website that contains information about the license. Typically, you make the name of the license a hyperlink. For example, if the notice is **Text under CC-BY-SA license** and CC-BY-SA is the name of the license, you would make CC-BY-SA a hyperlink.

Link and Text Attribution

The LinkAttribution and TextAttribution rules are typically used to identify the provider of the data. The `targetPropertyName` field identifies the field that the rule applies to.

To attribute the providers, include a line immediately following the content that the attributions apply to (for example, the targeted field). The line should be clearly labeled to indicate that the providers are the source of the data. For example, "Data from: en.wikipedia.org". For `LinkAttribution` rules, you must create a hyperlink to the provider's website.

The following shows an example that includes `LinkAttribution` and `TextAttribution` rules.



```
"contractualRules" : [{
    "_type" : "ContractualRules\/LinkAttribution",
    "targetPropertyName" : "description",
    "mustBeCloseToContent" : true,
    "text" : "en.wikipedia.org",
   "url" : "http:\/\/en.wikipedia.org\/wiki\/Peyton_Manning",
 },
 {
   "_type" : "ContractualRules\/LinkAttribution",
   "text" : "Wikipedia",
   "url" : "http:\/\/en.wikipedia.org\/wiki\/Peyton_Manning",
 },
 {
   "_type" : "ContractualRules\/TextAttribution",
   "text" : "STATS LLC © 2016"
 },
 {
    "_type" : "ContractualRules\/LinkAttribution",
   "text" : "Geni",
   "url" : "https:\/\/www.geni.com\/people\/Peyton...",
 },
 {
   "_type" : "ContractualRules\/LinkAttribution",
   "text" : "S-msn",
   "url" : "http:\/\/img.s-msn.com\/tenant\/amp\/entityid\/AA...",
 },
 {
   "_type" : "ContractualRules\/LinkAttribution",
   "text" : "Freebase",
   "url" : "http:\/\/www.freebase.com\/",
 }],
```

Peyton Williams Manning is a former American football quarterback who played 18 seasons in the National Football League. A five-time league MVP, he played for the Indianapolis Colts for 14 seasons between 1998 and 2011, and then with the Denver Broncos for four seasons from 2012 to 2015. Manning is a two-time Super Bowl winner, and also appeared in the game two other times, becoming the only quarterback to start the Super Bowl for two different franchises more than once each and the only starting quarterback to win a Super Bowl with two different franchises. He is a son of former NFL quarterback Archie Manning and older brother of New York Giants quarterback Eli Manning.

en.wikipedia.org · Text under CC-BY-SA license

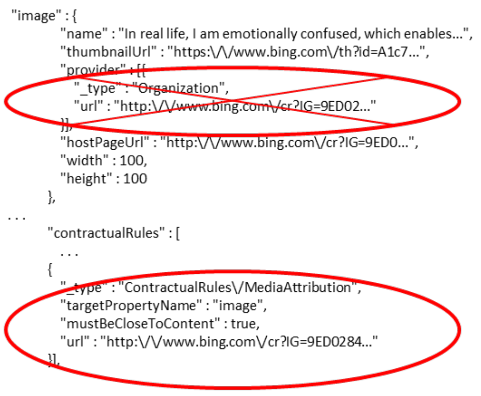
Wikipedia

Data from: Wikipedia · STATS LLC © 2016 · Geni · S-msn · Freebase

Media Attribution

If the entity includes an image and you display it, you must provide a click-through link to the provider's website. If the entity includes a MediaAttribution rule, use the rule's URL to create the click-through link. Otherwise, use the URL included in the image's `provider` field to create the click-through link.

The following shows an example that includes an image's `provider` field and contractual rules. Because the example includes the contractual rule, you will ignore the image's `provider` field and apply the `MediaAttribution` rule.



```
"image" : {
        "name" : "In real life, I am emotionally confused, which enables...",
        "thumbnailUrl" : "https:\/\/www.bing.com\/th?id=A1c7...",
        "provider" : [{
           ~~"_type" : "Organization",~~
           ~~"url" : "http:\/\/www.bing.com\/cr?IG=9ED02..."~~
        }],
        "hostPageUrl" : "http:\/\/www.bing.com\/cr?IG=9ED0...",
        "width" : 100,
        "height" : 100
      },
. . .
      "contractualRules" : [
        . . .
      {
        "_type" : "ContractualRules\/MediaAttribution",
        "targetPropertyName" : "image",
        "mustBeCloseToContent" : true,
        "url" : "http:\/\/www.bing.com\/cr?IG=9ED0284..."
      }],
```

Search or search-like experience

Just like with Bing Web Search API, the Entity Search API may only be used as a result of a direct user query or search, or as a result of an action within an app or experience that logically can be interpreted as a user's search request. For illustration purposes, the following are some examples of acceptable Search or Search-like experiences.

- User enters a query directly into a search box in an app

- User selects specific text or image and requests “more information” or “additional information”
- User asks a search bot about a particular topic
- User dwells on a particular object or entity in a visual search type scenario

  If you are not sure if your experience can be considered a search-like experience, it is recommended that you check with Microsoft.

Next steps

To get started quickly with your first request, see Making Your First Request.

Familiarize yourself with the Entities Search API Reference. The reference contains the headers and query parameters that you use to request search results. It also includes definitions of the response objects.

To improve your search box user experience, see Bing Autosuggest API. As the user enters their query term, you can call this API to get relevant query terms that were used by others.

Be sure to read Bing Use and Display Requirements so you don't break any of the rules about using the search results.

Entity Search endpoints

1/12/2018 • 1 min to read • Edit Online

The **Entity Search API** includes one endpoint.

Endpoint

To request entity search results, send a request to the following endpoint. Use the headers and URL parameters to define further specifications.

Endpoint `GET`:

```
https://api.cognitive.microsoft.com/bing/v7.0/entities
```

The following URL parameters are required:

- mkt. The market where the results come from.
- q. The entity search query.

Next steps

Bing Entity Search quickstarts

See also

Bing Entity Search overview API Reference

Quickstart for Microsoft Bing Entity Search API with C#

1/12/2018 • 2 min to read • Edit Online

This article shows you how to use the Bing Entity Search API with C#.

Prerequisites

You will need Visual Studio 2017 to run this code on Windows. (The free Community Edition will work.)

You must have a Cognitive Services API account with **Bing Entity Search API**. The free trial is sufficient for this quickstart. You need the access key provided when you activate your free trial, or you may use a paid subscription key from your Azure dashboard.

Search entities

To run this application, follow these steps.

1. Create a new C# project in your favorite IDE.
2. Add the code provided below.
3. Replace the `key` value with an access key valid for your subscription.
4. Run the program.

```
using System;
using System.Net.Http;
using System.Text;

namespace EntitySearchSample
{
    class Program
    {
        static string host = "https://api.cognitive.microsoft.com";
        static string path = "/bing/v7.0/entities";

        static string market = "en-US";

        // NOTE: Replace this example key with a valid subscription key.
        static string key = "ENTER KEY HERE";

        static string query = "italian restaurant near me";

        async static void Search()
        {
            HttpClient client = new HttpClient();
            client.DefaultRequestHeaders.Add("Ocp-Apim-Subscription-Key", key);

            string uri = host + path + "?mkt=" + market + "&q=" + System.Net.WebUtility.UrlEncode(query);

            HttpResponseMessage response = await client.GetAsync(uri);

            string contentString = await response.Content.ReadAsStringAsync();
            Console.WriteLine(JsonPrettyPrint(contentString));
        }

        static void Main(string[] args)
        {
            Search();
            Search();
            Console.ReadLine();
        }


        static string JsonPrettyPrint(string json)
        {
            if (string.IsNullOrEmpty(json))
                return string.Empty;

            json = json.Replace(Environment.NewLine, "").Replace("\t", "");

            StringBuilder sb = new StringBuilder();
            bool quote = false;
            bool ignore = false;
            int offset = 0;
            int indentLength = 3;

            foreach (char ch in json)
            {
                switch (ch)
                {
                    case '"':
                        if (!ignore) quote = !quote;
                        break;
                    case '\'':
                        if (quote) ignore = !ignore;
                        break;
                }

                if (quote)
                    sb.Append(ch);
                else
                {
                    switch (ch)
                    {
                        case '{':
                        case '[':
                            sb.Append(ch);
                            sb.Append(Environment.NewLine);
                            sb.Append(new string(' ', ++offset * indentLength));
                            break;
                        case '}':
                        case ']':
                            sb.Append(Environment.NewLine);
                            sb.Append(new string(' ', --offset * indentLength));
                            sb.Append(ch);
                            break;
                        case ',':
                            sb.Append(ch);
                            sb.Append(Environment.NewLine);
                            sb.Append(new string(' ', offset * indentLength));
                            break;
                        case ':':
                            sb.Append(ch);
                            sb.Append(' ');
                            break;
                        default:
                            if (ch != ' ') sb.Append(ch);
                            break;
                    }
                }
            }

            return sb.ToString().Trim();
        }

    }
}
```

Response

A successful response is returned in JSON, as shown in the following example:

```
{
  "_type": "SearchResponse",
  "queryContext": {
    "originalQuery": "italian restaurant near me",
    "askUserForLocation": true
  },
  "places": {
    "value": [
      {
        "_type": "LocalBusiness",
        "webSearchUrl": "https://www.bing.com/search?
q=Park+Place&filters=local_ypid:%22YN873x5786319842120194005%22&elv=AXXfrEiqqD9r3GuelwApulqDCgnOZrYZ*RB3VGaWfk8
gK7yMNsMKZ091jipuxw7sD8M5EX84K6nRW*6aYSd2s*n!ZICJHXshywvARqsAvOi4",
        "name": "Park Place",
        "url": "https://www.restaurantparkplace.com/",
        "entityPresentationInfo": {
          "entityScenario": "ListItem",
          "entityTypeHints": [
            "Place",
            "LocalBusiness"
          ]
        },
        "address": {
          "addressLocality": "Seattle",
          "addressRegion": "WA",
          "postalCode": "98112",
          "addressCountry": "US",
          "neighborhood": "Madison Park"
        },
        "telephone": "(206) 453-5867"
      },
      {
        "_type": "LocalBusiness",
        "webSearchUrl": "https://www.bing.com/search?
q=Pasta+and+Company&filters=local_ypid:%22YN873x2257558900374394159%22&elv=AXXfrEiqqD9r3GuelwApulqDCgnOZrYZ*RB3
VGaWfk8gK7yMNsMKZ091jipuxw7sD8M5EX84K6nRW*6aYSd2s*n!ZICJHXshywvARqsAvOi4",
        "name": "Pasta and Company",
        "url": "http://www.pastaco.com/",
        "entityPresentationInfo": {
          "entityScenario": "ListItem",
          "entityTypeHints": [
            "Place",
            "LocalBusiness"
          ]
        },
        "address": {
          "addressLocality": "Seattle",
          "addressRegion": "WA",
          "postalCode": "98121",
          "addressCountry": "US",
          "neighborhood": ""
        },
        "telephone": "(206) 322-1644"
      },
      {
        "_type": "LocalBusiness",
        "webSearchUrl": "https://www.bing.com/search?q=Calozzi%27s+Cheesesteaks-
Italian&filters=local_ypid:%22YN925x222744375%22&elv=AXXfrEiqqD9r3GuelwApulqDCgnOZrYZ*RB3VGaWfk8gK7yMNsMKZ091ji
puxw7sD8M5EX84K6nRW*6aYSd2s*n!ZICJHXshywvARqsAvOi4",
        "name": "Calozzi's Cheesesteaks-Italian",
        "entityPresentationInfo": {
          "entityScenario": "ListItem",
          "entityTypeHints": [
            "Place",
            "LocalBusiness"
          ]
        },
        "address": {
          "addressLocality": "Bellevue",
          "addressRegion": "WA",
          "postalCode": "98008",
          "addressCountry": "US",
          "neighborhood": "Crossroads"
        },
        "telephone": "(425) 221-5116"
      },
      {
        "_type": "Restaurant",
        "webSearchUrl": "https://www.bing.com/search?
q=Princi&filters=local_ypid:%22YN873x3764731790710239496%22&elv=AXXfrEiqqD9r3GuelwApulqDCgnOZrYZ*RB3VGaWfk8gK7y
MNsMKZ091jipuxw7sD8M5EX84K6nRW*6aYSd2s*n!ZICJHXshywvARqsAvOi4",
        "name": "Princi",
        "url": "http://www.princi.com/",
        "entityPresentationInfo": {
          "entityScenario": "ListItem",
          "entityTypeHints": [
            "Place",
            "LocalBusiness",
            "Restaurant"
          ]
        },
        "address": {
          "addressLocality": "Seattle",
          "addressRegion": "WA",
          "postalCode": "98101",
          "addressCountry": "US",
          "neighborhood": "Capitol Hill"
        },
        "telephone": "(206) 624-0173"
      },
      {
        "_type": "Restaurant",
        "webSearchUrl": "https://www.bing.com/search?
q=Swedish+Ballard+Cafeteria&filters=local_ypid:%22YN873x9787543113095303180%22&elv=AXXfrEiqqD9r3GuelwApulqDCgnO
ZrYZ*RB3VGaWfk8gK7yMNsMKZ091jipuxw7sD8M5EX84K6nRW*6aYSd2s*n!ZICJHXshywvARqsAvOi4",
        "name": "Swedish Ballard Cafeteria",
        "url": "http://www.swedish.com/",
        "entityPresentationInfo": {
          "entityScenario": "ListItem",
          "entityTypeHints": [
            "Place",
            "LocalBusiness",
            "Restaurant"
          ]
        },
        "address": {
          "addressLocality": "Seattle",
          "addressRegion": "WA",
          "postalCode": "98107",
          "addressCountry": "US",
          "neighborhood": "Ballard"
        }
      }
    ]
  }
}
```

Back to top

Next steps

Bing Entity Search tutorial

See also

Bing Entity Search overview API Reference

Quickstart for Microsoft Bing Entity Search API with Java

1/12/2018 • 3 min to read • Edit Online

This article shows you how to use the Bing Entity Search API with Java.

Prerequisites

You will need JDK 7 or 8 to compile and run this code. You may use a Java IDE if you have a favorite, but a text editor will suffice.

You must have a Cognitive Services API account with **Bing Entity Search API**. The free trial is sufficient for this quickstart. You need the access key provided when you activate your free trial, or you may use a paid subscription key from your Azure dashboard.

Search entities

To run this application, follow these steps.

1. Create a new Java project in your favorite IDE.
2. Add the code provided below.
3. Replace the `key` value with an access key valid for your subscription.
4. Run the program.

```
import java.io.*;
import java.net.*;
import java.util.*;
import javax.net.ssl.HttpsURLConnection;

/*
 * Gson: https://github.com/google/gson
 * Maven info:
 *     groupId: com.google.code.gson
 *     artifactId: gson
 *     version: 2.8.1
 *
 * Once you have compiled or downloaded gson-2.8.1.jar, assuming you have placed it in the
 * same folder as this file (EntitySearch.java), you can compile and run this program at
 * the command line as follows.
 *
 * javac EntitySearch.java -cp .;gson-2.8.1.jar
 * java -cp .;gson-2.8.1.jar EntitySearch
 */
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import com.google.gson.JsonObject;
import com.google.gson.JsonParser;

public class EntitySearch {

// ***********************************************
// *** Update or verify the following values. ***
// ***********************************************

// Replace the subscriptionKey string value with your valid subscription key.
    static String subscriptionKey = "ENTER KEY HERE";
    static String host = "https://api.cognitive.microsoft.com";
    static String path = "/bing/v7.0/entities";

    static String mkt = "en-US";
    static String query = "italian restaurant near me";

    public static String search () throws Exception {
        String encoded_query = URLEncoder.encode (query, "UTF-8");
        String params = "?mkt=" + mkt + "&q=" + encoded_query;
        URL url = new URL (host + path + params);

        HttpsURLConnection connection = (HttpsURLConnection) url.openConnection();
        connection.setRequestMethod("GET");
        connection.setRequestProperty("Ocp-Apim-Subscription-Key", subscriptionKey);
        connection.setDoOutput(true);

        StringBuilder response = new StringBuilder ();
        BufferedReader in = new BufferedReader(
        new InputStreamReader(connection.getInputStream()));
        String line;
        while ((line = in.readLine()) != null) {
            response.append(line);
        }
        in.close();

        return response.toString();
    }

    public static String prettify (String json_text) {
        JsonParser parser = new JsonParser();
        JsonObject json = parser.parse(json_text).getAsJsonObject();
        Gson gson = new GsonBuilder().setPrettyPrinting().create();
        return gson.toJson(json);
    }

    public static void main(String[] args) {
        try {
            String response = search ();
            System.out.println (prettify (response));
        }
        catch (Exception e) {
            System.out.println (e);
        }
    }
}
```

Response

A successful response is returned in JSON, as shown in the following example:

```
{
  "_type": "SearchResponse",
  "queryContext": {
    "originalQuery": "italian restaurant near me",
    "askUserForLocation": true
  },
  "places": {
    "value": [
      {
        "_type": "LocalBusiness",
        "webSearchUrl": "https://www.bing.com/search?
q=Park+Place&filters=local_ypid:%22YN873x5786319842120194005%22&elv=AXXfrEiqqD9r3GuelwApulqDCgnOZrYZ*RB3VGaWfk8
gK7yMNsMKZ091jipuxw7sD8M5EX84K6nRW*6aYSd2s*n!ZICJHXshywvARqsAvOi4",
        "name": "Park Place",
        "url": "https://www.restaurantparkplace.com/",
        "entityPresentationInfo": {
          "entityScenario": "ListItem",
          "entityTypeHints": [
            "Place",
            "LocalBusiness"
          ]
        },
        "address": {
          "addressLocality": "Seattle",
          "addressRegion": "WA",
          "postalCode": "98112",
          "addressCountry": "US",
          "neighborhood": "Madison Park"
        },
        "telephone": "(206) 453-5867"
      },
      {
        "_type": "LocalBusiness",
        "webSearchUrl": "https://www.bing.com/search?
q=Pasta+and+Company&filters=local_ypid:%22YN873x2257558900374394159%22&elv=AXXfrEiqqD9r3GuelwApulqDCgnOZrYZ*RB3
VGaWfk8gK7yMNsMKZ091jipuxw7sD8M5EX84K6nRW*6aYSd2s*n!ZICJHXshywvARqsAvOi4",
        "name": "Pasta and Company",
        "url": "http://www.pastaco.com/",
        "entityPresentationInfo": {
          "entityScenario": "ListItem",
          "entityTypeHints": [
            "Place",
            "LocalBusiness"
          ]
        },
        "address": {
          "addressLocality": "Seattle",
          "addressRegion": "WA",
          "postalCode": "98121",
          "addressCountry": "US",
          "neighborhood": ""
        },
        "telephone": "(206) 322-1644"
      },
      {
        "_type": "LocalBusiness",
        "webSearchUrl": "https://www.bing.com/search?q=Calozzi%27s+Cheesesteaks-
Italian&filters=local_ypid:%22YN925x222744375%22&elv=AXXfrEiqqD9r3GuelwApulqDCgnOZrYZ*RB3VGaWfk8gK7yMNsMKZ091ji
puxw7sD8M5EX84K6nRW*6aYSd2s*n!ZICJHXshywvARqsAvOi4",
        "name": "Calozzi's Cheesesteaks-Italian",
        "entityPresentationInfo": {
          "entityScenario": "ListItem",
          "entityTypeHints": [
            "Place",
            "LocalBusiness"
          ]
        },
        "address": {
          "addressLocality": "Bellevue",
          "addressRegion": "WA",
          "postalCode": "98008",
          "addressCountry": "US",
          "neighborhood": "Crossroads"
        },
        "telephone": "(425) 221-5116"
      },
      {
        "_type": "Restaurant",
        "webSearchUrl": "https://www.bing.com/search?
q=Princi&filters=local_ypid:%22YN873x3764731790710239496%22&elv=AXXfrEiqqD9r3GuelwApulqDCgnOZrYZ*RB3VGaWfk8gK7y
MNsMKZ091jipuxw7sD8M5EX84K6nRW*6aYSd2s*n!ZICJHXshywvARqsAvOi4",
        "name": "Princi",
        "url": "http://www.princi.com/",
        "entityPresentationInfo": {
          "entityScenario": "ListItem",
          "entityTypeHints": [
            "Place",
            "LocalBusiness",
            "Restaurant"
          ]
        },
        "address": {
          "addressLocality": "Seattle",
          "addressRegion": "WA",
          "postalCode": "98101",
          "addressCountry": "US",
          "neighborhood": "Capitol Hill"
        },
        "telephone": "(206) 624-0173"
      },
      {
        "_type": "Restaurant",
        "webSearchUrl": "https://www.bing.com/search?
q=Swedish+Ballard+Cafeteria&filters=local_ypid:%22YN873x9787543113095303180%22&elv=AXXfrEiqqD9r3GuelwApulqDCgnO
ZrYZ*RB3VGaWfk8gK7yMNsMKZ091jipuxw7sD8M5EX84K6nRW*6aYSd2s*n!ZICJHXshywvARqsAvOi4",
        "name": "Swedish Ballard Cafeteria",
        "url": "http://www.swedish.com/",
        "entityPresentationInfo": {
          "entityScenario": "ListItem",
          "entityTypeHints": [
            "Place",
            "LocalBusiness",
            "Restaurant"
          ]
        },
        "address": {
          "addressLocality": "Seattle",
          "addressRegion": "WA",
          "postalCode": "98107",
          "addressCountry": "US",
          "neighborhood": "Ballard"
        }
      }
    ]
  }
}
```

Back to top

Next steps

Bing Entity Search tutorial

See also

Bing Entity Search overview API Reference

Quickstart for Microsoft Bing Entity Search API with Node.JS

1/12/2018 • 2 min to read • Edit Online

This article shows you how to use the Bing Entity Search API with Node.JS.

Prerequisites

You will need Node.js 6 to run this code.

You must have a Cognitive Services API account with **Bing Entity Search API**. The free trial is sufficient for this quickstart. You need the access key provided when you activate your free trial, or you may use a paid subscription key from your Azure dashboard.

Search entities

To run this application, follow these steps.

1. Create a new Node.JS project in your favorite IDE.
2. Add the code provided below.
3. Replace the `key` value with an access key valid for your subscription.
4. Run the program.

```
'use strict';

let https = require ('https');

// **********************************************
// *** Update or verify the following values. ***
// **********************************************

// Replace the subscriptionKey string value with your valid subscription key.
let subscriptionKey = 'ENTER KEY HERE';

let host = 'api.cognitive.microsoft.com';
let path = '/bing/v7.0/entities';

let mkt = 'en-US';
let q = 'italian restaurant near me';

let params = '?mkt=' + mkt + '&q=' + encodeURI(q);

let response_handler = function (response) {
    let body = '';
    response.on ('data', function (d) {
        body += d;
    });
    response.on ('end', function () {
        let body_ = JSON.parse (body);
        let body__ = JSON.stringify (body_, null, '  ');
        console.log (body__);
    });
    response.on ('error', function (e) {
        console.log ('Error: ' + e.message);
    });
};

let Search = function () {
    let request_params = {
        method : 'GET',
        hostname : host,
        path : path + params,
        headers : {
            'Ocp-Apim-Subscription-Key' : subscriptionKey,
        }
    };

    let req = https.request (request_params, response_handler);
    req.end ();
}

Search ();
```

Response

A successful response is returned in JSON, as shown in the following example:

```
{
  "_type": "SearchResponse",
  "queryContext": {
    "originalQuery": "italian restaurant near me",
    "askUserForLocation": true
  },
  "places": {
    "value": [
      {
        "_type": "LocalBusiness",
        "webSearchUrl": "https://www.bing.com/search?
q=Park+Place&filters=local_ypid:%22YN873x5786319842120194005%22&elv=AXXfrEiqqD9r3GuelwApulqDCgnOZrYZ*RB3VGaWfk8
```

```
gK7yMNsMKZ091jipuxw7sD8M5EX84K6nRW*6aYSd2s*n!ZICJHXshywvARqsAvOi4",
        "name": "Park Place",
        "url": "https://www.restaurantparkplace.com/",
        "entityPresentationInfo": {
          "entityScenario": "ListItem",
          "entityTypeHints": [
            "Place",
            "LocalBusiness"
          ]
        },
        "address": {
          "addressLocality": "Seattle",
          "addressRegion": "WA",
          "postalCode": "98112",
          "addressCountry": "US",
          "neighborhood": "Madison Park"
        },
        "telephone": "(206) 453-5867"
      },
      {
        "_type": "LocalBusiness",
        "webSearchUrl": "https://www.bing.com/search?
q=Pasta+and+Company&filters=local_ypid:%22YN873x2257558900374394159%22&elv=AXXfrEiqqD9r3GuelwApulqDCgnOZrYZ*RB3
VGaWfk8gK7yMNsMKZ091jipuxw7sD8M5EX84K6nRW*6aYSd2s*n!ZICJHXshywvARqsAvOi4",
        "name": "Pasta and Company",
        "url": "http://www.pastaco.com/",
        "entityPresentationInfo": {
          "entityScenario": "ListItem",
          "entityTypeHints": [
            "Place",
            "LocalBusiness"
          ]
        },
        "address": {
          "addressLocality": "Seattle",
          "addressRegion": "WA",
          "postalCode": "98121",
          "addressCountry": "US",
          "neighborhood": ""
        },
        "telephone": "(206) 322-1644"
      },
      {
        "_type": "LocalBusiness",
        "webSearchUrl": "https://www.bing.com/search?q=Calozzi%27s+Cheesesteaks-
Italian&filters=local_ypid:%22YN925x222744375%22&elv=AXXfrEiqqD9r3GuelwApulqDCgnOZrYZ*RB3VGaWfk8gK7yMNsMKZ091ji
puxw7sD8M5EX84K6nRW*6aYSd2s*n!ZICJHXshywvARqsAvOi4",
        "name": "Calozzi's Cheesesteaks-Italian",
        "entityPresentationInfo": {
          "entityScenario": "ListItem",
          "entityTypeHints": [
            "Place",
            "LocalBusiness"
          ]
        },
        "address": {
          "addressLocality": "Bellevue",
          "addressRegion": "WA",
          "postalCode": "98008",
          "addressCountry": "US",
          "neighborhood": "Crossroads"
        },
        "telephone": "(425) 221-5116"
      },
      {
        "_type": "Restaurant",
        "webSearchUrl": "https://www.bing.com/search?
q=Princi&filters=local_ypid:%22YN873x3764731790710239496%22&elv=AXXfrEiqqD9r3GuelwApulqDCgnOZrYZ*RB3VGaWfk8gK7y
MNsMKZ091jipuxw7sD8M5EX84K6nRW*6aYSd2s*n!ZICJHXshywvARqsAvOi4",
        "name": "Princi",
        "url": "http://www.princi.com/",
        "entityPresentationInfo": {
          "entityScenario": "ListItem",
          "entityTypeHints": [
            "Place",
            "LocalBusiness",
            "Restaurant"
          ]
        },
        "address": {
          "addressLocality": "Seattle",
          "addressRegion": "WA",
          "postalCode": "98101",
          "addressCountry": "US",
          "neighborhood": "Capitol Hill"
        },
        "telephone": "(206) 624-0173"
      },
      {
        "_type": "Restaurant",
        "webSearchUrl": "https://www.bing.com/search?
q=Swedish+Ballard+Cafeteria&filters=local_ypid:%22YN873x9787543113095303180%22&elv=AXXfrEiqqD9r3GuelwApulqDCgnO
ZrYZ*RB3VGaWfk8gK7yMNsMKZ091jipuxw7sD8M5EX84K6nRW*6aYSd2s*n!ZICJHXshywvARqsAvOi4",
        "name": "Swedish Ballard Cafeteria",
        "url": "http://www.swedish.com/",
        "entityPresentationInfo": {
          "entityScenario": "ListItem",
          "entityTypeHints": [
            "Place",
            "LocalBusiness",
            "Restaurant"
          ]
        },
        "address": {
          "addressLocality": "Seattle",
          "addressRegion": "WA",
          "postalCode": "98107",
          "addressCountry": "US",
          "neighborhood": "Ballard"
        }
      }
    ]
  }
}
```

Back to top

Next steps

Bing Entity Search tutorial

See also

Bing Entity Search overview API Reference

Quickstart for Microsoft Bing Entity Search API with Python

1/12/2018 • 2 min to read • Edit Online

This article shows you how to use the Bing Entity Search API with Python.

Prerequisites

You will need Python 3.x to run this code.

You must have a Cognitive Services API account with **Bing Entity Search API**. The free trial is sufficient for this quickstart. You need the access key provided when you activate your free trial, or you may use a paid subscription key from your Azure dashboard.

Search entities

To run this application, follow these steps.

1. Create a new Python project in your favorite IDE.
2. Add the code provided below.
3. Replace the `key` value with an access key valid for your subscription.
4. Run the program.

```
# -*- coding: utf-8 -*-

import http.client, urllib.parse
import json

# **********************************************
# *** Update or verify the following values. ***
# **********************************************

# Replace the subscriptionKey string value with your valid subscription key.
subscriptionKey = 'ENTER KEY HERE'

host = 'api.cognitive.microsoft.com'
path = '/bing/v7.0/entities'

mkt = 'en-US'
query = 'italian restaurants near me'

params = '?mkt=' + mkt + '&q=' + urllib.parse.quote (query)

def get_suggestions ():
    headers = {'Ocp-Apim-Subscription-Key': subscriptionKey}
    conn = http.client.HTTPSConnection (host)
    conn.request ("GET", path + params, None, headers)
    response = conn.getresponse ()
    return response.read ()

result = get_suggestions ()
print (json.dumps(json.loads(result), indent=4))
```

Response

A successful response is returned in JSON, as shown in the following example:

```
{
  "_type": "SearchResponse",
  "queryContext": {
    "originalQuery": "italian restaurant near me",
    "askUserForLocation": true
  },
  "places": {
    "value": [
      {
        "_type": "LocalBusiness",
        "webSearchUrl": "https://www.bing.com/search?
q=Park+Place&filters=local_ypid:%22YN873x5786319842120194005%22&elv=AXXfrEiqqD9r3GuelwApulqDCgnOZrYZ*RB3VGaWfk8
gK7yMNsMKZ091jipuxw7sD8M5EX84K6nRW*6aYSd2s*n!ZICJHXshywvARqsAvOi4",
        "name": "Park Place",
        "url": "https://www.restaurantparkplace.com/",
        "entityPresentationInfo": {
          "entityScenario": "ListItem",
          "entityTypeHints": [
            "Place",
            "LocalBusiness"
          ]
        },
        "address": {
          "addressLocality": "Seattle",
          "addressRegion": "WA",
          "postalCode": "98112",
          "addressCountry": "US",
          "neighborhood": "Madison Park"
        },
        "telephone": "(206) 453-5867"
      },
      {
        "_type": "LocalBusiness",
        "webSearchUrl": "https://www.bing.com/search?
q=Pasta+and+Company&filters=local_ypid:%22YN873x2257558900374394159%22&elv=AXXfrEiqqD9r3GuelwApulqDCgnOZrYZ*RB3
VGaWfk8gK7yMNsMKZ091jipuxw7sD8M5EX84K6nRW*6aYSd2s*n!ZICJHXshywvARqsAvOi4",
        "name": "Pasta and Company",
        "url": "http://www.pastaco.com/",
        "entityPresentationInfo": {
          "entityScenario": "ListItem",
          "entityTypeHints": [
            "Place",
            "LocalBusiness"
          ]
        },
        "address": {
          "addressLocality": "Seattle",
          "addressRegion": "WA",
          "postalCode": "98121",
          "addressCountry": "US",
          "neighborhood": ""
        },
        "telephone": "(206) 322-1644"
      },
      {
        "_type": "LocalBusiness",
        "webSearchUrl": "https://www.bing.com/search?q=Calozzi%27s+Cheesesteaks-
Italian&filters=local_ypid:%22YN925x222744375%22&elv=AXXfrEiqqD9r3GuelwApulqDCgnOZrYZ*RB3VGaWfk8gK7yMNsMKZ091ji
puxw7sD8M5EX84K6nRW*6aYSd2s*n!ZICJHXshywvARqsAvOi4",
        "name": "Calozzi's Cheesesteaks-Italian",
        "entityPresentationInfo": {
          "entityScenario": "ListItem",
          "entityTypeHints": [
            "Place",
            "LocalBusiness"
          ]
        },
        "address": {
          "addressLocality": "Bellevue",
          "addressRegion": "WA",
          "postalCode": "98008",
          "addressCountry": "US",
          "neighborhood": "Crossroads"
        },
        "telephone": "(425) 221-5116"
      },
      {
        "_type": "Restaurant",
        "webSearchUrl": "https://www.bing.com/search?
q=Princi&filters=local_ypid:%22YN873x3764731790710239496%22&elv=AXXfrEiqqD9r3GuelwApulqDCgnOZrYZ*RB3VGaWfk8gK7y
MNsMKZ091jipuxw7sD8M5EX84K6nRW*6aYSd2s*n!ZICJHXshywvARqsAvOi4",
        "name": "Princi",
        "url": "http://www.princi.com/",
        "entityPresentationInfo": {
          "entityScenario": "ListItem",
          "entityTypeHints": [
            "Place",
            "LocalBusiness",
            "Restaurant"
          ]
        },
        "address": {
          "addressLocality": "Seattle",
          "addressRegion": "WA",
          "postalCode": "98101",
          "addressCountry": "US",
          "neighborhood": "Capitol Hill"
        },
        "telephone": "(206) 624-0173"
      },
      {
        "_type": "Restaurant",
        "webSearchUrl": "https://www.bing.com/search?
q=Swedish+Ballard+Cafeteria&filters=local_ypid:%22YN873x9787543113095303180%22&elv=AXXfrEiqqD9r3GuelwApulqDCgnO
ZrYZ*RB3VGaWfk8gK7yMNsMKZ091jipuxw7sD8M5EX84K6nRW*6aYSd2s*n!ZICJHXshywvARqsAvOi4",
        "name": "Swedish Ballard Cafeteria",
        "url": "http://www.swedish.com/",
        "entityPresentationInfo": {
          "entityScenario": "ListItem",
          "entityTypeHints": [
            "Place",
            "LocalBusiness",
            "Restaurant"
          ]
        },
        "address": {
          "addressLocality": "Seattle",
          "addressRegion": "WA",
          "postalCode": "98107",
          "addressCountry": "US",
          "neighborhood": "Ballard"
        }
      }
    ]
  }
}
```

Back to top

Next steps

Bing Entity Search tutorial

See also

Bing Entity Search overview API Reference

Quickstart for Microsoft Bing Entity Search API with PHP

1/12/2018 • 2 min to read • Edit Online

This article shows you how to use the Bing Entity Search API with PHP.

Prerequisites

You will need PHP 5.6.x to run this code.

You must have a Cognitive Services API account with **Bing Entity Search API**. The free trial is sufficient for this quickstart. You need the access key provided when you activate your free trial, or you may use a paid subscription key from your Azure dashboard.

Search entities

To run this application, follow these steps.

1. Create a new PHP project in your favorite IDE.
2. Add the code provided below.
3. Replace the `key` value with an access key valid for your subscription.
4. Run the program.

```php
<?php

// NOTE: Be sure to uncomment the following line in your php.ini file.
// ;extension=php_openssl.dll

// **********************************************
// *** Update or verify the following values. ***
// **********************************************

// Replace the subscriptionKey string value with your valid subscription key.
$subscriptionKey = 'ENTER KEY HERE';

$host = "https://api.cognitive.microsoft.com";
$path = "/bing/v7.0/entities";

$mkt = "en-US";
$query = "italian restaurants near me";

function search ($host, $path, $key, $mkt, $query) {

    $params = '?mkt=' . $mkt . '&q=' . urlencode ($query);

    $headers = "Ocp-Apim-Subscription-Key: $key\r\n";

    // NOTE: Use the key 'http' even if you are making an HTTPS request. See:
    // http://php.net/manual/en/function.stream-context-create.php
    $options = array (
        'http' => array (
            'header' => $headers,
            'method' => 'GET'
        )
    );
    $context  = stream_context_create ($options);
    $result = file_get_contents ($host . $path . $params, false, $context);
    return $result;
}

$result = search ($host, $path, $subscriptionKey, $mkt, $query);

echo json_encode (json_decode ($result), JSON_PRETTY_PRINT);
?>
```

Response

A successful response is returned in JSON, as shown in the following example:

```json
{
  "_type": "SearchResponse",
  "queryContext": {
    "originalQuery": "italian restaurant near me",
    "askUserForLocation": true
  },
  "places": {
    "value": [
      {
        "_type": "LocalBusiness",
        "webSearchUrl": "https://www.bing.com/search?
q=Park+Place&filters=local_ypid:%22YN873x5786319842120194005%22&elv=AXXfrEiqqD9r3GuelwApulqDCgnOZrYZ*RB3VGaWfk8
gK7yMNsMKZ091jipuxw7sD8M5EX84K6nRW*6aYSd2s*n!ZICJHXshywvARqsAvOi4",
        "name": "Park Place",
        "url": "https://www.restaurantparkplace.com/",
        "entityPresentationInfo": {
          "entityScenario": "ListItem",
          "entityTypeHints": [
            "Place",
            "LocalBusiness"
          ]
        },
        "address": {
          "addressLocality": "Seattle",
          "addressRegion": "WA",
          "postalCode": "98112",
          "addressCountry": "US",
          "neighborhood": "Madison Park"
        },
        "telephone": "(206) 453-5867"
      },
      {
        "_type": "LocalBusiness",
        "webSearchUrl": "https://www.bing.com/search?
q=Pasta+and+Company&filters=local_ypid:%22YN873x2257558900374394159%22&elv=AXXfrEiqqD9r3GuelwApulqDCgnOZrYZ*RB3
VGaWfk8gK7yMNsMKZ091jipuxw7sD8M5EX84K6nRW*6aYSd2s*n!ZICJHXshywvARqsAvOi4",
        "name": "Pasta and Company",
        "url": "http://www.pastaco.com/",
        "entityPresentationInfo": {
          "entityScenario": "ListItem",
          "entityTypeHints": [
            "Place",
            "LocalBusiness"
          ]
        },
        "address": {
          "addressLocality": "Seattle",
          "addressRegion": "WA",
          "postalCode": "98121",
          "addressCountry": "US",
          "neighborhood": ""
        },
        "telephone": "(206) 322-1644"
      },
      {
        "_type": "LocalBusiness",
        "webSearchUrl": "https://www.bing.com/search?q=Calozzi%27s+Cheesesteaks-
Italian&filters=local_ypid:%22YN925x222744375%22&elv=AXXfrEiqqD9r3GuelwApulqDCgnOZrYZ*RB3VGaWfk8gK7yMNsMKZ091ji
puxw7sD8M5EX84K6nRW*6aYSd2s*n!ZICJHXshywvARqsAvOi4",
        "name": "Calozzi's Cheesesteaks-Italian",
        "entityPresentationInfo": {
          "entityScenario": "ListItem",
          "entityTypeHints": [
            "Place",
            "LocalBusiness"
          ]
        },
        "address": {
          "addressLocality": "Bellevue",
          "addressRegion": "WA",
          "postalCode": "98008",
          "addressCountry": "US",
          "neighborhood": "Crossroads"
        },
        "telephone": "(425) 221-5116"
      },
      {
        "_type": "Restaurant",
        "webSearchUrl": "https://www.bing.com/search?
q=Princi&filters=local_ypid:%22YN873x3764731790710239496%22&elv=AXXfrEiqqD9r3GuelwApulqDCgnOZrYZ*RB3VGaWfk8gK7y
MNsMKZ091jipuxw7sD8M5EX84K6nRW*6aYSd2s*n!ZICJHXshywvARqsAvOi4",
        "name": "Princi",
        "url": "http://www.princi.com/",
        "entityPresentationInfo": {
          "entityScenario": "ListItem",
          "entityTypeHints": [
            "Place",
            "LocalBusiness",
            "Restaurant"
          ]
        },
        "address": {
          "addressLocality": "Seattle",
          "addressRegion": "WA",
          "postalCode": "98101",
          "addressCountry": "US",
          "neighborhood": "Capitol Hill"
        },
        "telephone": "(206) 624-0173"
      },
      {
        "_type": "Restaurant",
        "webSearchUrl": "https://www.bing.com/search?
q=Swedish+Ballard+Cafeteria&filters=local_ypid:%22YN873x9787543113095303180%22&elv=AXXfrEiqqD9r3GuelwApulqDCgnO
ZrYZ*RB3VGaWfk8gK7yMNsMKZ091jipuxw7sD8M5EX84K6nRW*6aYSd2s*n!ZICJHXshywvARqsAvOi4",
        "name": "Swedish Ballard Cafeteria",
        "url": "http://www.swedish.com/",
        "entityPresentationInfo": {
          "entityScenario": "ListItem",
          "entityTypeHints": [
            "Place",
            "LocalBusiness",
            "Restaurant"
          ]
        },
        "address": {
          "addressLocality": "Seattle",
          "addressRegion": "WA",
          "postalCode": "98107",
          "addressCountry": "US",
          "neighborhood": "Ballard"
        }
      }
    ]
  }
}
```

Back to top

Next steps

Bing Entity Search tutorial

See also

Bing Entity Search overview API Reference

Quickstart for Microsoft Bing Entity Search API with Ruby

1/12/2018 • 2 min to read • Edit Online

This article shows you how to use the Bing Entity Search API with Ruby.

Prerequisites

You will need Ruby 2.4 or later to run this code.

You must have a Cognitive Services API account with **Bing Entity Search API**. The free trial is sufficient for this quickstart. You need the access key provided when you activate your free trial, or you may use a paid subscription key from your Azure dashboard.

Search entities

To run this application, follow these steps.

1. Create a new Ruby project in your favorite IDE.
2. Add the code provided below.
3. Replace the `key` value with an access key valid for your subscription.
4. Run the program.

```
require 'net/https'
require 'cgi'
require 'json'

# **********************************************
# *** Update or verify the following values. ***
# **********************************************

# Replace the subscriptionKey string value with your valid subscription key.
subscriptionKey = 'ENTER KEY HERE'

host = 'https://api.cognitive.microsoft.com'
path = '/bing/v7.0/entities'

mkt = 'en-US'
query = 'italian restaurants near me'

params = '?mkt=' + mkt + '&q=' + CGI.escape(query)
uri = URI (host + path + params)

request = Net::HTTP::Get.new(uri)
request['Ocp-Apim-Subscription-Key'] = subscriptionKey

response = Net::HTTP.start(uri.host, uri.port, :use_ssl => uri.scheme == 'https') do |http|
    http.request (request)
end

puts JSON::pretty_generate (JSON (response.body))
```

Response

A successful response is returned in JSON, as shown in the following example:

```
{
  "_type": "SearchResponse",
  "queryContext": {
    "originalQuery": "italian restaurant near me",
    "askUserForLocation": true
  },
  "places": {
    "value": [
      {
        "_type": "LocalBusiness",
        "webSearchUrl": "https://www.bing.com/search?
q=Park+Place&filters=local_ypid:%22YN873x5786319842120194005%22&elv=AXXfrEiqqD9r3GuelwApulqDCgnOZrYZ*RB3VGaWfk8
gK7yMNsMKZ091jipuxw7sD8M5EX84K6nRW*6aYSd2s*n!ZICJHXshywvARqsAvOi4",
        "name": "Park Place",
        "url": "https://www.restaurantparkplace.com/",
        "entityPresentationInfo": {
          "entityScenario": "ListItem",
          "entityTypeHints": [
            "Place",
            "LocalBusiness"
          ]
        },
        "address": {
          "addressLocality": "Seattle",
          "addressRegion": "WA",
          "postalCode": "98112",
          "addressCountry": "US",
          "neighborhood": "Madison Park"
        },
        "telephone": "(206) 453-5867"
      },
      {
        "_type": "LocalBusiness",
        "webSearchUrl": "https://www.bing.com/search?
q=Pasta+and+Company&filters=local_ypid:%22YN873x2257558900374394159%22&elv=AXXfrEiqqD9r3GuelwApulqDCgnOZrYZ*RB3
VGaWfk8gK7yMNsMKZ091jipuxw7sD8M5EX84K6nRW*6aYSd2s*n!ZICJHXshywvARqsAvOi4",
        "name": "Pasta and Company",
        "url": "http://www.pastaco.com/",
        "entityPresentationInfo": {
          "entityScenario": "ListItem",
          "entityTypeHints": [
            "Place",
            "LocalBusiness"
          ]
        },
        "address": {
          "addressLocality": "Seattle",
          "addressRegion": "WA",
          "postalCode": "98121",
          "addressCountry": "US",
          "neighborhood": ""
        },
        "telephone": "(206) 322-1644"
      },
      {
        "_type": "LocalBusiness",
        "webSearchUrl": "https://www.bing.com/search?q=Calozzi%27s+Cheesesteaks-
Italian&filters=local_ypid:%22YN925x222744375%22&elv=AXXfrEiqqD9r3GuelwApulqDCgnOZrYZ*RB3VGaWfk8gK7yMNsMKZ091ji
puxw7sD8M5EX84K6nRW*6aYSd2s*n!ZICJHXshywvARqsAvOi4",
        "name": "Calozzi's Cheesesteaks-Italian",
        "entityPresentationInfo": {
          "entityScenario": "ListItem",
          "entityTypeHints": [
            "Place",
            "LocalBusiness"
          ]
        },
        "address": {
          "addressLocality": "Bellevue",
          "addressRegion": "WA",
          "postalCode": "98008",
          "addressCountry": "US",
          "neighborhood": "Crossroads"
        },
        "telephone": "(425) 221-5116"
      },
      {
        "_type": "Restaurant",
        "webSearchUrl": "https://www.bing.com/search?
q=Princi&filters=local_ypid:%22YN873x3764731790710239496%22&elv=AXXfrEiqqD9r3GuelwApulqDCgnOZrYZ*RB3VGaWfk8gK7y
MNsMKZ091jipuxw7sD8M5EX84K6nRW*6aYSd2s*n!ZICJHXshywvARqsAvOi4",
        "name": "Princi",
        "url": "http://www.princi.com/",
        "entityPresentationInfo": {
          "entityScenario": "ListItem",
          "entityTypeHints": [
            "Place",
            "LocalBusiness",
            "Restaurant"
          ]
        },
        "address": {
          "addressLocality": "Seattle",
          "addressRegion": "WA",
          "postalCode": "98101",
          "addressCountry": "US",
          "neighborhood": "Capitol Hill"
        },
        "telephone": "(206) 624-0173"
      },
      {
        "_type": "Restaurant",
        "webSearchUrl": "https://www.bing.com/search?
q=Swedish+Ballard+Cafeteria&filters=local_ypid:%22YN873x9787543113095303180%22&elv=AXXfrEiqqD9r3GuelwApulqDCgnO
ZrYZ*RB3VGaWfk8gK7yMNsMKZ091jipuxw7sD8M5EX84K6nRW*6aYSd2s*n!ZICJHXshywvARqsAvOi4",
        "name": "Swedish Ballard Cafeteria",
        "url": "http://www.swedish.com/",
        "entityPresentationInfo": {
          "entityScenario": "ListItem",
          "entityTypeHints": [
            "Place",
            "LocalBusiness",
            "Restaurant"
          ]
        },
        "address": {
          "addressLocality": "Seattle",
          "addressRegion": "WA",
          "postalCode": "98107",
          "addressCountry": "US",
          "neighborhood": "Ballard"
        }
      }
    ]
  }
}
```

Back to top

Next steps

Bing Entity Search tutorial

See also

Bing Entity Search overview API Reference

Bing Search SDK preview

1/26/2018 • 1 min to read • Edit Online

The Bing Entity Search API samples include scenarios that:

1. Search for entity such as Tom Cruise and get rich information.
2. Handle disambiguation of terms for queries with possibly multiple intents.
3. Search for a local entity such as a restaurant and get rich information around it.
4. Search for local businesses such as restaurants and get rich information.
5. Trigger a bad request and error handling.

The Bing Search SDKs make web search functionality readily accessible in the following programming languages:

- Get started with .NET samples
  - NuGet package
  - See also .NET libraries for definitions and dependencies.
- Get started with Node.js samples
  - See also Node.js libraries for definitions and dependencies.
- Get started with Java samples
  - See also Java libraries for definitions and dependencies.
- Get started with Python samples
  - See also Python libraries for definitions and dependencies.

SDK samples for each language include a ReadMe file with details about prerequisites and installing/running the samples.

Entity Search SDK C# quickstart

3/8/2018 • 4 min to read • Edit Online

The Bing Entity Search API contains the functionality of the REST API for entity search and parsing the results.

Application dependencies

To set up a console application using the Bing Entity Search SDK, browse to the `Manage NuGet Packages` option from the Solution Explorer in Visual Studio. Add the `Microsoft.Azure.CognitiveServices.Search.EntitySearch` package.

Installing the NuGet Entity Search package will also install dependencies, including the following assemblies:

- Microsoft.Rest.ClientRuntime
- Microsoft.Rest.ClientRuntime.Azure
- Newtonsoft.Json

Entity Search client

To create an instance of the `EntitySearchAPI` client, add using directives:

```
using Microsoft.Azure.CognitiveServices.Search.EntitySearch;
using Microsoft.Azure.CognitiveServices.Search.EntitySearch.Models;
```

Then, instantiate the client:

```
var client = new EntitySearchAPI(new ApiKeyServiceClientCredentials("YOUR-ACCESS-KEY"));
```

Use the client to search with a query text:

```
var entityData = client.Entities.Search(query: "Satya Nadella");
```

Parse the results of previous query:

```
if (entityData?.Entities?.Value?.Count > 0)
{
    // find the entity that represents the dominant one
    var mainEntity = entityData.Entities.Value.Where(thing => thing.EntityPresentationInfo.EntityScenario ==
EntityScenario.DominantEntity).FirstOrDefault();

    if (mainEntity != null)
    {
        Console.WriteLine("Searched for \"Satya Nadella\" and found a dominant entity with this description:");
        Console.WriteLine(mainEntity.Description);
    }
    else
    {
        Console.WriteLine("Couldn't find main entity Satya Nadella!");
    }
}
else
{
    Console.WriteLine("Didn't see any data..");
}
```

Complete console application

The following console application looks up a single entity on query "Satya Nadella" and prints out a short description.

```
using System;
using System.Linq;
using System.Text;
using Microsoft.Azure.CognitiveServices.Search.EntitySearch;
using Microsoft.Azure.CognitiveServices.Search.EntitySearch.Models;
using Newtonsoft.Json;

namespace EntitySrchSDK
{
    class Program
    {
        static void Main(string[] args)
        {
            var client = new EntitySearchAPI(new 
ApiKeyServiceClientCredentials("19aa718a79d6444daaa415981d9f54ad"));

            DominantEntityLookup(client);

            // Include the following methods to use queries defined under the headings following this example.
            //HandlingDisambiguation(client);
            //RestaurantLookup(client);
            //MultipleRestaurantLookup(client);
            //Error(client);

            Console.WriteLine("Any key to exit...");
            Console.ReadKey();
        }

        public static void DominantEntityLookup(EntitySearchAPI client)
        {
            try
            {
                var entityData = client.Entities.Search(query: "Satya Nadella");

                if (entityData?.Entities?.Value?.Count > 0)
                {
                    // find the entity that represents the dominant one
                    var mainEntity = entityData.Entities.Value.Where(thing => 
thing.EntityPresentationInfo.EntityScenario == EntityScenario.DominantEntity).FirstOrDefault();

                    if (mainEntity != null)
                    {
                        Console.WriteLine("Searched for \"Satya Nadella\" and found a dominant entity with this 
description:");
                        Console.WriteLine(mainEntity.Description);
                    }
                    else
                    {
                        Console.WriteLine("Couldn't find main entity Satya Nadella!");
                    }
                }
                else
                {
                    Console.WriteLine("Didn't see any data..");
                }
            }
            catch (ErrorResponseException ex)
            {
                Console.WriteLine("Encountered exception. " + ex.Message);
            }
        }
    }
}
```

Ambiguous results

The following code handles disambiguation of results for an ambiguous query "William Gates".

```
        public static void HandlingDisambiguation(EntitySearchAPI client)
        {
            try
            {
                var entityData = client.Entities.Search(query: "william gates");

                if (entityData?.Entities?.Value?.Count > 0)
                {
                    // find the entity that represents the dominant one
                    var mainEntity = entityData.Entities.Value.Where(thing =>
thing.EntityPresentationInfo.EntityScenario == EntityScenario.DominantEntity).FirstOrDefault();
                    var disambigEntities = entityData.Entities.Value.Where(thing =>
thing.EntityPresentationInfo.EntityScenario == EntityScenario.DisambiguationItem).ToList();

                    if (mainEntity != null)
                    {
                        Console.WriteLine("Searched for \"William Gates\" and found a dominant entity with type
hint \"{0}\" with this description:", mainEntity.EntityPresentationInfo.EntityTypeDisplayHint);
                        Console.WriteLine(mainEntity.Description);
                    }
                    else
                    {
                        Console.WriteLine("Couldn't find a reliable dominant entity for William Gates!");
                    }

                    if (disambigEntities?.Count > 0)
                    {
                        Console.WriteLine();
                        Console.WriteLine("This query is pretty ambiguous and can be referring to multiple
things. Did you mean one of these: ");

                        var sb = new StringBuilder();

                        foreach (var disambig in disambigEntities)
                        {
                            sb.AppendFormat(", or {0} the {1}", disambig.Name,
disambig.EntityPresentationInfo.EntityTypeDisplayHint);
                        }

                        Console.WriteLine(sb.ToString().Substring(5) + "?");
                    }
                    else
                    {
                        Console.WriteLine("We didn't find any disambiguation items for William Gates, so we
must be certain what you're talking about!");
                    }
                }
                else
                {
                    Console.WriteLine("Didn't see any data..");
                }
            }
            catch (ErrorResponseException ex)
            {
                Console.WriteLine("Encountered exception. " + ex.Message);
            }
        }
```

EntityData places

The following code looks up a single store "Microsoft Store" and prints out its phone number.

```
        public static void StoreLookup(EntitySearchAPI client)
        {
            try
            {
                var entityData = client.Entities.Search(query: "microsoft store");

                if (entityData?.Places?.Value?.Count > 0)
                {
                    // Some local entities will be places, others won't be. Depending on the data you want, try
to cast to the appropriate schema.
                    // In this case, the item being returned is technically a store, but the Place schema has
the data we want (telephone)
                    var store = entityData.Places.Value.FirstOrDefault() as Place;

                    if (store != null)
                    {
                        Console.WriteLine("Searched for \"Microsoft Store\" and found a store with this phone
number:");
                        Console.WriteLine(store.Telephone);
                    }
                    else
                    {
                        Console.WriteLine("Couldn't find a place!");
                    }
                }
                else
                {
                    Console.WriteLine("Didn't see any data..");
                }
            }
            catch (ErrorResponseException ex)
            {
                Console.WriteLine("Encountered exception. " + ex.Message);
            }
        }
```

EntityScenario list

The following code looks up a list of "Seattle restaurants" and prints their names and phone numbers.

```
        public static void MultipleRestaurantLookup(EntitySearchAPI client)
         {
            try
            {
                var restaurants = client.Entities.Search(query: "seattle restaurants");

                if (restaurants?.Places?.Value?.Count > 0)
                {
                    // get all the list items that relate to this query
                    var listItems = restaurants.Places.Value.Where(thing =>
thing.EntityPresentationInfo.EntityScenario == EntityScenario.ListItem).ToList();

                    if (listItems?.Count > 0)
                    {
                        var sb = new StringBuilder();

                        foreach (var item in listItems)
                        {
                            var place = item as Place;

                            if (place == null)
                            {
                                Console.WriteLine("Unexpectedly found something that isn't a place named \"
{0}\"", item.Name);

                                continue;
                            }

                            sb.AppendFormat(",{0} ({1}) ", place.Name, place.Telephone);
                        }

                        Console.WriteLine("Ok, we found these places: ");
                        Console.WriteLine(sb.ToString().Substring(1));
                    }
                    else
                    {
                        Console.WriteLine("Couldn't find any relevant results for \"seattle restaurants\"");
                    }
                }
                else
                {
                    Console.WriteLine("Didn't see any data..");
                }
            }
            catch (ErrorResponseException ex)
            {
                Console.WriteLine("Encountered exception. " + ex.Message);
            }
         }
```

Error results

The following code triggers a bad request and shows how to read the error response.

```
        public static void Error(EntitySearchAPI client)
        {
            try
            {
                var entityData = client.Entities.Search(query: "satya nadella", market: "no-ty");
            }
            catch (ErrorResponseException ex)
            {
                // The status code of the error should be a good indication of what occurred. However, if you'd
like more details, you can dig into the response.
                // Please note that depending on the type of error, the response schema might be different, so
you aren't guaranteed a specific error response schema.
                Console.WriteLine("Exception occurred, status code {0} with reason {1}.",
ex.Response.StatusCode, ex.Response.ReasonPhrase);

                // if you'd like more descriptive information (if available), attempt to parse the content as
an ErrorResponse
                var issue = JsonConvert.DeserializeObject<ErrorResponse>(ex.Response.Content);

                if (issue != null && issue.Errors?.Count > 0)
                {
                    if (issue.Errors[0].SubCode == ErrorSubCode.ParameterInvalidValue)
                    {
                        Console.WriteLine("Turns out the issue is parameter \"{0}\" has an invalid value \"
{1}\". Detailed message is \"{2}\"", issue.Errors[0].Parameter, issue.Errors[0].Value,
issue.Errors[0].Message);
                    }
                }
            }
        }
```

Next steps

Cognitive services .NET SDK samples

Entity Search SDK Node quickstart (preview)

2/21/2018 • 1 min to read • Edit Online

The Bing Entity Search SDK contains the functionality of the REST API for entity queries and parsing results.

Application dependencies

To set up a console application using the Bing Entity Search SDK, run `npm install azure-cognitiveservices-entitysearch` in your development environment.

Entity Search client

Get a Cognitive Services access key under *Search*. Create an instance of the `CognitiveServicesCredentials`:

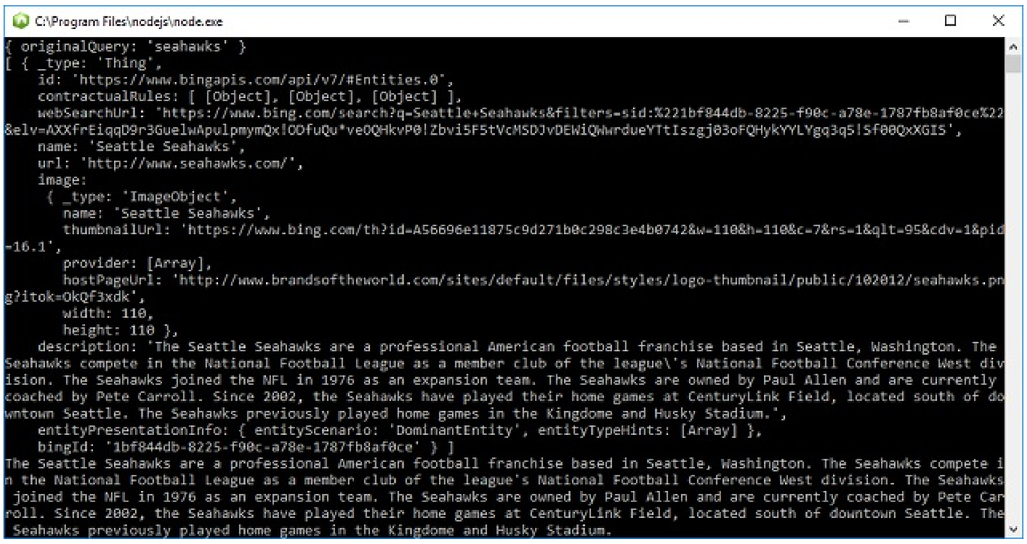
```
const CognitiveServicesCredentials = require('ms-rest-azure').CognitiveServicesCredentials;
let credentials = new CognitiveServicesCredentials('YOUR-ACCESS-KEY');
```

Then, instantiate the client, and search for results:

```
const EntitySearchAPIClient = require('azure-cognitiveservices-entitysearch');

let entitySearchApiClient = new EntitySearchAPIClient(credentials);

entitySearchApiClient.entitiesOperations.search('seahawks').then((result) => {
    console.log(result.queryContext);
    console.log(result.entities.value);
    console.log(result.entities.value[0].description);
}).catch((err) => {
    throw err;
});
```

The code prints `result.value` items to the console without parsing any text. The results, if any per category, will include:

- _type: 'Thing'
- _type: 'ImageObject'



```
{ originalQuery: 'seahawks' }
[ { _type: 'Thing',
    id: 'https://www.bingapis.com/api/v7/#Entities.0',
    contractualRules: [ [Object], [Object], [Object] ],
    webSearchUrl: 'https://www.bing.com/search?q=Seattle+Seahawks&filters=sid:%221bf844db-8225-f90c-a78e-1787fb8af0ce%22
&elv=AXXfrEiqqD9r3GuelwApulpmymQx!ODfuQu*veOQHkvP0!Zbvi5F5tVcMSDJvDEWiQWwrdueYTtIszgj03oFQHykYYLYgq3q5!Sf00QxXGIS',
    name: 'Seattle Seahawks',
    url: 'http://www.seahawks.com/',
    image:
     { _type: 'ImageObject',
       name: 'Seattle Seahawks',
       thumbnailUrl: 'https://www.bing.com/th?id=A56696e11875c9d271b0c298c3e4b0742&w=110&h=110&c=7&rs=1&qlt=95&cdv=1&pid
=16.1',
       provider: [Array],
       hostPageUrl: 'http://www.brandsoftheworld.com/sites/default/files/styles/logo-thumbnail/public/102012/seahawks.pn
g?itok=OkQf3xdk',
       width: 110,
       height: 110 },
    description: 'The Seattle Seahawks are a professional American football franchise based in Seattle, Washington. The
Seahawks compete in the National Football League as a member club of the league\'s National Football Conference West div
ision. The Seahawks joined the NFL in 1976 as an expansion team. The Seahawks are owned by Paul Allen and are currently
coached by Pete Carroll. Since 2002, the Seahawks have played their home games at CenturyLink Field, located south of do
wntown Seattle. The Seahawks previously played home games in the Kingdome and Husky Stadium.',
    entityPresentationInfo: { entityScenario: 'DominantEntity', entityTypeHints: [Array] },
    bingId: '1bf844db-8225-f90c-a78e-1787fb8af0ce' } ]
The Seattle Seahawks are a professional American football franchise based in Seattle, Washington. The Seahawks compete i
n the National Football League as a member club of the league's National Football Conference West division. The Seahawks
 joined the NFL in 1976 as an expansion team. The Seahawks are owned by Paul Allen and are currently coached by Pete Car
roll. Since 2002, the Seahawks have played their home games at CenturyLink Field, located south of downtown Seattle. The
 Seahawks previously played home games in the Kingdome and Husky Stadium.
```

Next steps

Cognitive services Node.js SDK samples

Entity Search SDK Python quickstart

3/8/2018 • 3 min to read • Edit Online

The Entity Search SDK contains the functionality of the REST API for web queries and parsing results.

Application dependencies

If you don't already have it, install Python. The SDK is compatible with Python 2.7, 3.3, 3.4, 3.5, and 3.6.

The general recommendation for Python development is to use a virtual environment. Install and initialize the virtual environment with the venv module. You must install virtualenv for Python 2.7.

```
python -m venv mytestenv
```

Install Bing Entity Search SDK dependencies:

```
cd mytestenv
python -m pip install azure-cognitiveservices-search-entitysearch
```

Entity Search client

Get a Cognitive Services access key under *Search*. Add imports:

```
from azure.cognitiveservices.search.entitysearch import EntitySearchAPI
from azure.cognitiveservices.search.entitysearch.models import Place, ErrorResponseException
from msrest.authentication import CognitiveServicesCredentials

subscription_key = "YOUR-SUBSCRIPTION-KEY"
```

Create an instance of the `CognitiveServicesCredentials`. Instantiate the client:

```
client = EntitySearchAPI(CognitiveServicesCredentials(subscription_key))
```

Search for a single entity (Gibralter) and print out a short description:

```
entity_data = client.entities.search(query="Gibralter")

if entity_data.entities.value:
    # find the entity that represents the dominant one

    main_entities = [entity for entity in entity_data.entities.value
                     if entity.entity_presentation_info.entity_scenario == "DominantEntity"]

    if main_entities:
        print('Searched for "Gibralter" and found a dominant entity with this description:')
        print(main_entities[0].description)
    else:
        print("Couldn't find main entity Gibralter!")

else:
    print("Didn't see any data..")
```

Search and handle disambiguation results for an ambiguous query (William Gates).

```
def handling_disambiguation(subscription_key):

    client = EntitySearchAPI(CognitiveServicesCredentials(subscription_key))

    try:
        entity_data = client.entities.search(query="william gates")

        if entity_data.entities.value:
            # find the entity that represents the dominant one

            main_entities = [entity for entity in entity_data.entities.value
                             if entity.entity_presentation_info.entity_scenario == "DominantEntity"]

            disambig_entities = [entity for entity in entity_data.entities.value
                                 if entity.entity_presentation_info.entity_scenario == "DisambiguationItem"]

            if main_entities:
                main_entity = main_entities[0]
                type_hint = main_entity.entity_presentation_info.entity_type_display_hint

                print('Searched for "William Gates" and found a dominant entity {}with this
description:'.format(
                    '"with type hint "{}" '.format(type_hint) if type_hint else ''))
                print(main_entity.description)
            else:
                print("Couldn't find a reliable dominant entity for William Gates!")

            if disambig_entities:
                print("\nThis query is pretty ambiguous and can be referring to multiple things. Did you mean
one of these:")
                suggestions = []
                for disambig_entity in disambig_entities:
                    suggestions.append("{} the {}".format(disambig_entity.name,
disambig_entity.entity_presentation_info.entity_type_display_hint))
                print(", or ".join(suggestions))
            else:
                print("We didn't find any disambiguation items for William Gates, so we must be certain what
you're talking about!")

        else:
            print("Didn't see any data..")

    except Exception as err:
        print("Encountered exception. {}".format(err))
```

Search for a single store (Microsoft Store) and print out its phone number.

```
def store_lookup(subscription_key):

    client = EntitySearchAPI(CognitiveServicesCredentials(subscription_key))

    try:
        entity_data = client.entities.search(query="microsoft store")

        if entity_data.places.value:

            store = entity_data.places.value[0]

            # Some local entities will be places, others won't be. Depending on what class contains the data
you want, you can check
            # using isinstance one of the class, or try to get the attribute and handle the exception (EAFP
principle).
            # The recommended Python way is usually EAFP (see https://docs.python.org/3/glossary.html)
            # In this case, the item being returned is technically a store, but the Place schema has the data
we want (telephone)

            # Pythonic approach : EAFP "Easier to ask for forgiveness than permission"
            try:
                telephone = store.telephone
                print('Searched for "Microsoft Store" and found a store with this phone number:')
                print(telephone)
            except AttributeError:
                print("Couldn't find a place!")

            # More cross language approach
            if isinstance(store, Place):
                print('Searched for "Microsoft Store" and found a store with this phone number:')
                print(store.telephone)
            else:
                print("Couldn't find a place!")


        else:
            print("Didn't see any data..")

    except Exception as err:
        print("Encountered exception. {}".format(err))
```

Search for a list of restaurants (Seattle restaurants) and print their names and phone numbers.

```
def multiple_restaurant_lookup(subscription_key):

    client = EntitySearchAPI(CognitiveServicesCredentials(subscription_key))

    try:
        restaurants = client.entities.search(query="seattle restaurants")

        if restaurants.places.value:

            # get all the list items that relate to this query
            list_items = [entity for entity in restaurants.places.value
                          if entity.entity_presentation_info.entity_scenario == "ListItem"]

            if list_items:

                suggestions = []
                for place in list_items:
                    # Pythonic approach : EAFP "Easier to ask for forgiveness than permission"
                    # see https://docs.python.org/3/glossary.html
                    try:
                        suggestions.append("{} ({})".format(place.name, place.telephone))
                    except AttributeError:
                        print("Unexpectedly found something that isn\'t a place named '{}'", place.name)

                print("Ok, we found these places: ")
                print(", ".join(suggestions))

            else:
                print("Couldn't find any relevant results for \"seattle restaurants\"")

        else:
            print("Didn't see any data..")

    except Exception as err:
        print("Encountered exception. {}".format(err))
```

Trigger a bad request and read the error response.

```
def error(subscription_key):

    client = EntitySearchAPI(CognitiveServicesCredentials(subscription_key))

    try:
        entity_data = client.entities.search(query="satya nadella", market="no-ty")
    except ErrorResponseException as err:
        # The status code of the error should be a good indication of what occurred. However, if you'd like
more details, you can dig into the response.
        # Please note that depending on the type of error, the response schema might be different, so you
aren't guaranteed a specific error response schema.

        print("Exception occurred, status code {} with reason {}.\n".format(err.response.status_code, err))

        # if you'd like more descriptive information (if available)
        if err.error.errors:
            print("This is the errors I have:")
            for error in err.error.errors:
                print("Parameter \"{}\" has an invalid value \"{}\". SubCode is \"{}\". Detailed message is \"
{}\"".format(error.parameter, error.value, error.sub_code, error.message))
        else:
            print("There was no details on the error.")
```

Next steps

Cognitive Services Python SDK samples

Bing Entity Search SDK Java quickstart

3/8/2018 • 4 min to read • Edit Online

The Bing Entity Search SDK provides the REST API functionality for entity queries and parsing results.

Application dependencies

Get a Cognitive Services access key under **Search**. Install the Bing Entity Search SDK dependencies by using Maven, Gradle, or another dependency management system. The Maven POM file requires the declaration:

```
<dependencies>
  <dependency>
      <groupId>com.microsoft.azure.cognitiveservices</groupId>
      <artifactId>azure-cognitiveservices-entitysearch</artifactId>
      <version>0.0.1-beta-SNAPSHOT</version>
  </dependency>
</dependencies>
```

Entity Search client

Add imports to the class implementation.

```
import com.microsoft.azure.cognitiveservices.entitysearch.*;
import com.microsoft.azure.cognitiveservices.entitysearch.implementation.EntitySearchAPIImpl;
import com.microsoft.azure.cognitiveservices.entitysearch.implementation.SearchResponseInner;
import com.microsoft.rest.credentials.ServiceClientCredentials;
import okhttp3.Interceptor;
import okhttp3.OkHttpClient;
import okhttp3.Request;
import okhttp3.Response;

import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
```

Implement the **EntitySearchAPIImpl** client, which requires an instance of the **ServiceClientCredentials** class.

```
public static EntitySearchAPIImpl getClient(final String subscriptionKey) {
    return new EntitySearchAPIImpl("https://api.cognitive.microsoft.com/bing/v7.0/",
            new ServiceClientCredentials() {
                @Override
                public void applyCredentialsFilter(OkHttpClient.Builder builder) {
                    builder.addNetworkInterceptor(
                            new Interceptor() {
                                @Override
                                public Response intercept(Interceptor.Chain chain) throws IOException {
                                    Request request = null;
                                    Request original = chain.request();
                                    // Request customization: add request headers
                                    Request.Builder requestBuilder = original.newBuilder()
                                            .addHeader("Ocp-Apim-Subscription-Key", subscriptionKey);
                                    request = requestBuilder.build();
                                    return chain.proceed(request);
                                }
                            });
                }
            });
}
```

Search for the single entity "Satya Nadella" and print a short description.

```
public static void dominantEntityLookup(final String subscriptionKey)
{
    try
    {
        EntitySearchAPIImpl client = getClient(subscriptionKey);
        SearchResponseInner entityData = client.entities().search(
                "satya nadella", null, null, null, null, null, null, "en-us", null, null, SafeSearch.STRICT,
null);

        if (entityData.entities().value().size() > 0)
        {
            // Find the entity that represents the dominant entity
            List<Thing> entrys = entityData.entities().value();
            Thing dominateEntry = null;
            for(Thing thing : entrys) {
                if(thing.entityPresentationInfo().entityScenario() == EntityScenario.DOMINANT_ENTITY) {
                    System.out.println("\r\nSearched for \"Satya Nadella\" and found a dominant entity with
this description:");
                    System.out.println(thing.description());
                    break;
                }
            }

            if(dominateEntry == null)
            {
                 System.out.println("Couldn't find main entity Satya Nadella!");
            }
        }
        else
        {
             System.out.println("Didn't see any data..");
        }
    }
    catch (ErrorResponseException ex)
    {
         System.out.println("Encountered exception. " + ex.getLocalizedMessage());
    }
}
```

Search for "William Gates" and handle disambiguation results for the ambiguous query.

```
public static void handlingDisambiguation(String subscriptionKey)
{
    try
    {
        EntitySearchAPIImpl client = getClient(subscriptionKey);
        SearchResponseInner entityData = client.entities().search(
                "william gates", null, null, null, null, null, null, "en-us", null, null, SafeSearch.STRICT,
null);
        if (entityData.entities().value().size() > 0)
        {
            // Find the entity that represents the dominant entity
            List<Thing> entrys = entityData.entities().value();
            Thing dominateEntry = null;
            List<Thing> disambigEntities = new ArrayList<Thing>();
            for(Thing thing : entrys) {
                if(thing.entityPresentationInfo().entityScenario() == EntityScenario.DOMINANT_ENTITY) {
                    System.out.println("\r\nSearched for \"William Gates\" and found a dominant entity with
this description:");
                    System.out.println(String.format("Searched for \"William Gates\" and found a dominant
entity with type hint \"%s\" with this description:",
                            thing.entityPresentationInfo().entityTypeDisplayHint()));
                    System.out.println(thing.description());
                }

                if(thing.entityPresentationInfo().entityScenario() == EntityScenario.DISAMBIGUATION_ITEM) {
                    disambigEntities.add(thing);
                    System.out.println("Searched for \"William Gates\" and found a dominant entity with this
description:");
                    System.out.println(thing.description());
                }

            }

            if (dominateEntry == null)
            {
                 System.out.println("Couldn't find a reliable dominant entity for William Gates!");
            }

            if (disambigEntities.size() > 0)
            {
                 System.out.println();
                 System.out.println("This query is pretty ambiguous and can be referring to multiple things.
Did you mean one of these: ");

                StringBuilder sb = new StringBuilder();

                for (Thing disambig : disambigEntities)
                {
                    sb.append(String.format(", or %s the %s", disambig.name(),
disambig.entityPresentationInfo().entityTypeDisplayHint()));
                }

                 System.out.println(sb.toString().substring(5) + "?");
            }
            else
            {
                 System.out.println("We didn't find any disambiguation items for William Gates, so we must be
certain what you're talking about!");
            }
        }
        else
        {
             System.out.println("Didn't see any data..");
        }
    }
    catch (ErrorResponseException ex)
    {
         System.out.println("Encountered exception. " + ex.getLocalizedMessage());
    }
}
```

Search for a single store with the query "Microsoft Store" and print the phone number for the result.

```
public static void storeLookup(String subscriptionKey)
{
    try
    {
        EntitySearchAPIImpl client = getClient(subscriptionKey);
        SearchResponseInner entityData = client.entities().search(
                "Microsoft Store", null, null, null, null, null, null, "en-us", null, null, SafeSearch.STRICT,
null);

        if (entityData.places() != null && entityData.places().value().size() > 0)
        {
            // Some local entities are places, others are not. Depending on the data that you want, try to cast
to the appropriate schema.
            // In this case, the returned item is technically a store, but the Place schema has the data that
we want (telephone).
            Place store = (Place)entityData.places().value().get(0);

            if (store != null)
            {
                 System.out.println("\r\nSearched for \"Microsoft Store\" and found a store with this phone
number:");
                 System.out.println(store.telephone());
            }
            else
            {
                 System.out.println("Couldn't find a place!");
            }
        }
        else
        {
             System.out.println("Didn't see any data..");
        }
    }
    catch (ErrorResponseException ex)
    {
         System.out.println("Encountered exception. " + ex.getLocalizedMessage());
    }
}
```

Search for a list of restaurants with the query "Seattle restaurants." Print the names and phone numbers for the results.

```
public static void multipleRestaurantLookup(String subscriptionKey)
{
    try
    {
        EntitySearchAPIImpl client = getClient(subscriptionKey);
        SearchResponseInner restaurants = client.entities().search(
                "Seattle restaurants", null, null, null, null, null, null, "en-us", null, null,
SafeSearch.STRICT, null);

        System.out.println("\r\nSearched for \"multiple restaurants\"");
        if (restaurants.places() != null && restaurants.places().value().size() > 0)
        {
            List<Thing> listItems = new ArrayList<Thing>();
            for(Thing place : restaurants.places().value()) {
                if (place.entityPresentationInfo().entityScenario() == EntityScenario.LIST_ITEM) {
                    listItems.add(place);
                }
            }

            if (listItems.size() > 0)
            {
                StringBuilder sb = new StringBuilder();

                for (Thing item : listItems)
                {
                    Place place = (Place)item;
                    if (place == null)
                    {
                         System.out.println(String.format("Unexpectedly found something that isn't a place
named \"%s\"", place.name()));
                        continue;
                    }

                    sb.append(String.format(",%s (%s) ", place.name(), place.telephone()));
                }

                 System.out.println("Ok, we found these places: ");
                 System.out.println(sb.toString().substring(1));
            }
            else
            {
                 System.out.println("Couldn't find any relevant results for \"seattle restaurants\"");
            }
        }
        else
        {
             System.out.println("Didn't see any data..");
        }
    }
    catch (ErrorResponseException ex)
    {
         System.out.println("Encountered exception. " + ex.getLocalizedMessage());
    }
}
```

Add the methods described in this article to a class with a main function for executing the code.

```
package entitySDK;
import com.microsoft.azure.cognitiveservices.entitysearch.*;

public class EntitySearchSDK {

    public static void main(String[] args) {

        dominantEntityLookup("YOUR-SUBSCRIPTION-KEY");
        handlingDisambiguation("YOUR-SUBSCRIPTION-KEY");
        restaurantLookup("YOUR-SUBSCRIPTION-KEY");
        multipleRestaurantLookup("YOUR-SUBSCRIPTION-KEY");

    }
    // Include the methods described in this article.
}
```

Next steps

Cognitive Services Java SDK samples

Bing Search API use and display requirements

7/7/2017 • 7 min to read • Edit Online

2017-10-01

These use and display requirements apply to your implementation of the content and associated information (for example, relationships, metadata and other signals) available through calls to the Bing Custom Search, Entity Search, Image Search, News Search, Video Search, Web Search, Spell Check, and Autosuggest APIs. Implementation details related to these requirements can be found in documentation for specific features and results.

1. Bing Spell Check and Bing Autosuggest API.

You must not:

- copy, store, or cache any data you receive from the Bing Spell Check or Bing Autosuggest APIs
- use data you receive from the Bing Spell Check or Bing Autosuggest APIs as part of any machine learning or similar algorithmic activity to train, evaluate, or improve new or existing services which you or third parties may offer.

2. Definitions

- "answer" refers to a category of results returned in a response. For example, a response from the Bing Web Search API may include answers in the categories of webpage results, image, video, and news;
- "response" means any and all answers and associated data received in response to a single call to a Search API;
- "result" refers to an item of information in an answer. For example, the set of data connected with a single news article is a result in a news answer.
- “Search APIs” means, collectively, the Bing Custom Search, Entity Search, Image Search, News Search, Video Search, and Web Search APIs.

3. Search APIs

The requirements in this Section 3 apply to the Search APIs.

A. Internet search experience. All data returned in responses may only be used in internet search experiences. An internet search experience means the content displayed, as applicable:

- is relevant and responsive to the end user's direct query or other indication of the user's search interest and intent (for example, user-indicated search query);
- helps users find and navigate to the sources of data (for example, the provided URLs are implemented as hyperlinks so the content or attribution is a clickable link conspicuously displayed with the data); or, in the case of Bing Entity Search API, visibly link to the bing.com URL provided in the response that enables the user to navigate to the search results for the relevant query on bing.com;
- includes multiple results for the end user to select from (for example, several results from the news answer are displayed, or all results if fewer than several are returned);
- is limited to an amount appropriate to serve the search purpose (for example, image thumbnails are thumbnail-sized in proportion to the user's display);
- includes visible indication to the end user that the content is Internet search results (for example, a statement that the content is "From the web"); and
- includes any other combination of measures appropriate to ensure your use of data received from the Search APIs does not violate any applicable laws or third-party rights (for example, if relying on a Creative Commons

license, complying with the applicable license terms). Consult your legal advisors to determine what measures may be appropriate.

The only exception to the internet search experience requirement is for URL discovery as described in Section 3E (Non-display URL discovery) below.

B. Restrictions. You must not:

- copy, store, or cache any data from responses (except retention to the extent permitted by the "Continuity of Service" section below);
- use data received from the Search APIs as part of any machine learning or similar algorithmic activity to train, evaluate, or improve new or existing services which you or third parties may offer.
- modify content of results (other than to reformat them in a way that does not violate any other requirement), unless required by law or agreed by Microsoft;
- omit attribution and URLs associated with result content;
- re-order, including by omission, results displayed in an answer when an order or ranking is provided (for the Bing Custom Search API, this does not apply to re-ordering implemented through the customsearch.ai portal), unless required by law or agreed by Microsoft;
- display other content within any part of a response in a way that would lead an end user to believe that the other content is part of the response;
- display advertising that is not provided by Microsoft on any page that displays any part of a response;
- display any advertising with responses (i) from the Bing Image, News or Video Search APIs; or (ii) that are filtered or limited primarily (or solely) to image, news and/or video results.

C. Branding.

- You may attribute each response (or portion of a response) displayed from the Bing Web, Image, News, and Video APIs to Microsoft as described in https://go.microsoft.com/fwlink/?linkid=833278, unless Microsoft specifies otherwise in writing for your use.
- You must not attribute responses (or portions of responses) displayed from the Bing Custom Search API to Microsoft, unless Microsoft specifies otherwise in writing for your particular use.

D. Transferring responses. If you enable a user to transferIf you enable a user to transfer a response from a Search API to another user, such as through a messaging app or social media posting, the following apply:

- Transferred responses must:
  - Consist of content that is unmodified from the content of the responses displayed to the transferring user (formatting changes are permissible);
  - Not include any data in metadata form;
  - For responses from the Bing Web, Image, News, and Video APIs, display language indicating the response was obtained through an internet search experience powered by Bing (for example, "Powered by Bing," "Learn more about this image on Bing," or using the Bing logo);
  - For responses from the Bing Custom Search API, display language indicating the response was obtained through an internet search experience (for example, "Learn more about this search result");
  - Prominently display the full query used to generate the response; and
  - Include a prominent link or similar attribution to the underlying source of the response, either directly or through the search engine (bing.com, m.bing.com or your custom search service, as applicable).
- You may not automate the transfer of responses. A transfer must be initiated by a user action clearly evidencing an intent to transfer a response.
- You may only enable a user to transfer responses that were displayed in response to the transferring user's query.

E. Continuity of service. You must not copy, store or cache any data from Search API responses. However, to enable continuity of service access and data rendering, you may retain results solely under the following conditions:

Device. You may enable an end user to retain results on a device for the lesser of (i) 24 hours from the time of the query or (ii) until an end user submits another query for updated results, provided that retained results may be used only:

- to enable the end user to access results previously returned to that end user on that device (for example, in case of service interruption); or
- to store results returned for your proactive query personalized in anticipation of the end user's needs based on that end user's signals (for example, in case of anticipated service interruption).

Server. You may retain results specific to a single end user securely on a server you control and display the retained results only:

- to enable the end user to access a historical report of results previously returned to that user in your solution, provided that the results may not be (i) retained for more than 21 days from the time of the end user's initial query and (ii) displayed in response to an end user's new or repeated query; or
- to store results returned for your proactive query personalized in anticipation of an end user's needs based on that end user's signals for the lesser of (i) 24 hours from the time of the query or (ii) until an end user submits another query for updated results.

Whenever retained, results for a specific user cannot be commingled with results for another user, i.e., the results of each user must be retained and delivered separately.

General. For all presentation of retained results, you must:

- include a clear, visible notice of the time the query was sent,
- present the user a button or similar means to re-query and obtain updated results,
- retain the Bing branding in the presentation of the results, and
- delete (and refresh with a new query if needed) the stored results within the timeframes specified.

F. Non-display URL discovery. You may only use search responses in a non-internet search experience for the sole purpose of discovering URLs of sources of information responsive to a query from your user or customer. You may copy such URLs in a report or similar response you provide (i) only to that user or customer, in response to that query and (ii) which includes significant additional valuable content relevant to the query. The requirements in sections 3A through 3E of these use and display requirements do not apply to this non-display use, except:

- You shall not cache, copy or store any data or content from, or derived from, the search response, other than the limited URL copying described above;
- You must ensure your use of data (including the URLs) received from the Search APIs does not violate any applicable laws or third-party rights; and
- You shall not use the data (including the URLs) received from the Search APIs as part of any search index or machine learning or similar algorithmic activity to create train, evaluate, or improve services which you or third parties may offer.

Resizing and cropping thumbnail images

7/7/2017 • 3 min to read • Edit Online

Some Bing responses include URLs to thumbnail images served by Bing. You may resize and crop the thumbnail images.

NOTE

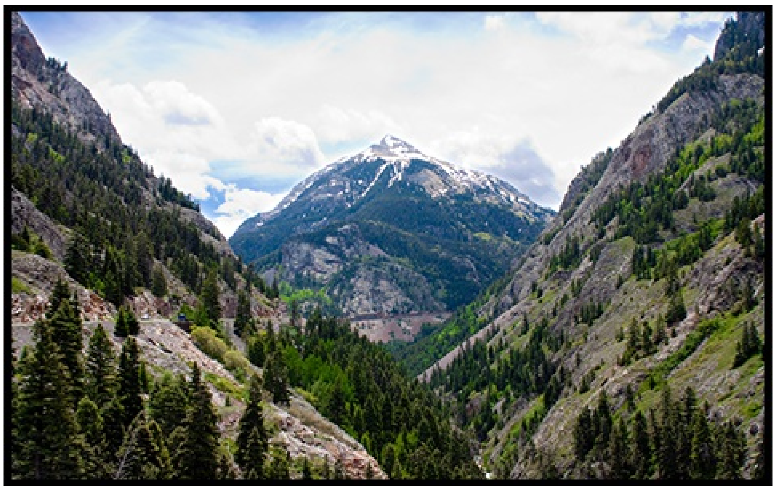
Ensure the size and cropping of the thumbnail provide a search scenario and respect third party rights, as required by Search API Use and Display Requirements.

To resize an image, include the w (width) and h (height) query parameters in the thumbnail's URL. Specify the width and height in pixels. For example:

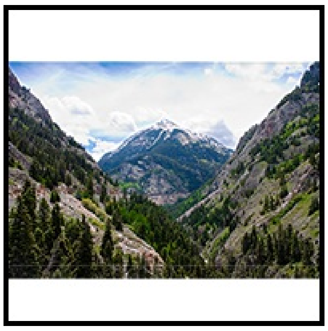
```
https://<host>/th?id=JN.5l3yzwy%2f%2fHj59U6XhssIQ&pid=Api&w=200&h=200
```

If you resize the image, its aspect ratio is maintained. To maintain the aspect ratio, white padding may be added to the border of the image. For example, if you resize a 480x359 image to 200x200 without cropping, the full width contains the image but the height contains 25 pixels of white padding at the top and bottom of the image. The same would be true if the image was 359x480 except the left and right borders would contain white padding. If you crop the image, white padding is not added.

The following picture shows the original size of a thumbnail image (480x300).



The following picture shows the image resized to 200x200. The aspect ratio is maintained and the top and bottom borders are padded with white (the black border is included to show the padding).

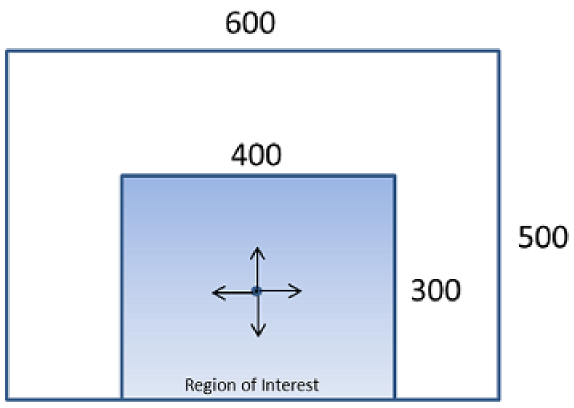


If you specify dimensions that are greater than the image's original width and height, the image is padded with white on the left and top borders.

To crop an image, include the c (crop) query parameter. The following are the possible values that you may specify.

- 4—Blind Ratio
- 7—Smart Ratio

If you request Smart Ratio cropping (c=7), the image is cropped from the center of the image's region of interest outward while maintaining the image's aspect ratio. The region of interest is the area of the image that Bing determines contains the most import parts. The following shows an example region of interest.

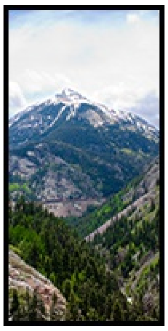


If you resize an image and request Smart Ratio cropping, the image is reduced to the requested size while maintaining the aspect ratio. The image is then cropped based on the resized dimensions. For example, if the resized width is less than or equal to the height, the image is cropped to the left and right of the center of the region of interest. Otherwise, the image is cropped to the top and bottom of the center of the region of interest.

The following shows the image reduced to 200x200 using Smart Ratio cropping.

The following shows the image reduced to 200x100 using Smart Ratio cropping.

The following shows the image reduced to 100x200 using Smart Ratio cropping.

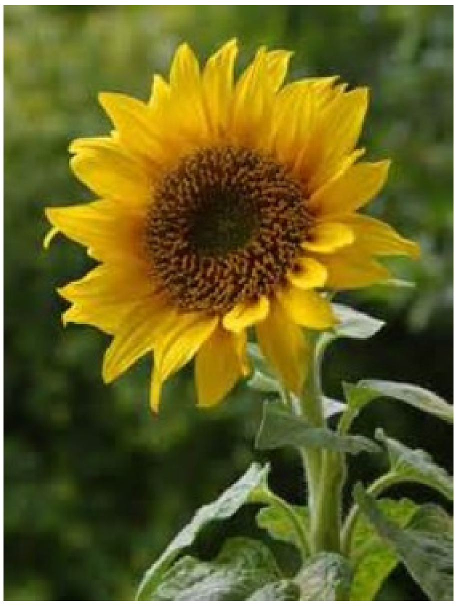


If Bing cannot determine the image's region of interest, Bing uses Blind Ratio cropping.

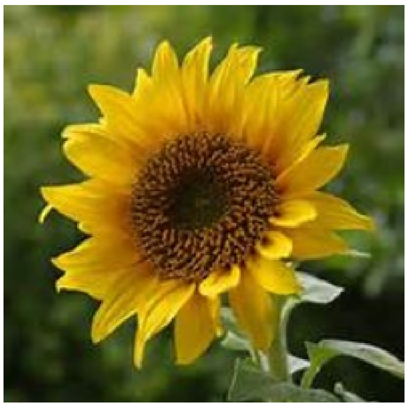
If you request Blind Ratio cropping (c=4), Bing uses the following rules to crop the image.

- If (Original Image Width / Original Image Height) < (Requested Image Width / Requested Image Height), the image is measured from top left corner and cropped at the bottom.
- If (Original Image Width / Original Image Height) > (Requested Image Width / Requested Image Height), the image is measured from the center and cropped to the left and right.

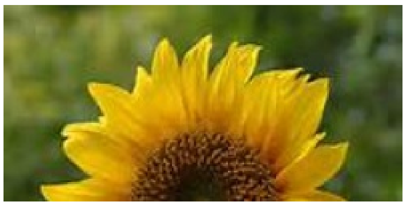
The following shows a portrait image that's 225x300.



The following shows the image reduced to 200x200 using Blind Ratio cropping. The image is measured from the top left corner resulting in the bottom part of the image being cropped.



The following shows the image reduced to 200x100 using Blind Ratio cropping. The image is measured from the top left corner resulting in the bottom part of the image being cropped.



The following shows the image reduced to 100x200 using Blind Ratio cropping. The image is measured from the center resulting in the left and right parts of the image being cropped.

Tutorial: Single-page web app

12/22/2017 • 18 min to read • Edit Online

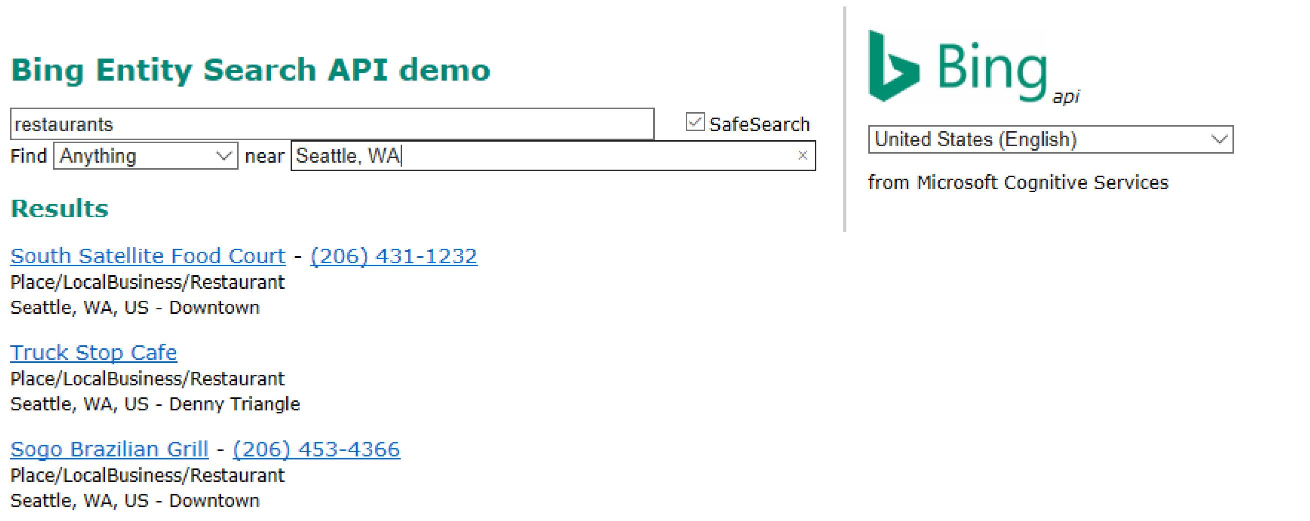
The Bing Entity Search API lets you search the Web for information about *entities* and *places*. You may request either kind of result, or both, in a given query. The definitions of places and entities are provided below.

Entities	Well-known people, places, and things that you find by name (Jimi Hendrix, the Space Needle, Microsoft Surface)
Places	Restaurants, hotels, and other local businesses that you find by name *or* by type (Italian restaurants)

In this tutorial, we build a single-page Web application that uses the Bing Entity Search API to display search results right in the page. The application includes HTML, CSS, and JavaScript components.

The API lets you prioritize results by location. In a mobile app, you can ask the device for its own location. In a Web app, you can use the `getPosition()` function. But this call works only in secure contexts, and it may not provide a precise location. Also, the user may want to search for entities near a location other than their own.

Our app therefore calls upon the Bing Maps service to obtain latitude and longitude from a user-entered location. The user can then enter the name of a landmark ("Space Needle") or a full or partial address ("New York City"), and the Bing Maps API provides the coordinates.


Bing Entity Search API demo

restaurants ☑ SafeSearch

Find Anything near Seattle, WA

Results

South Satellite Food Court - (206) 431-1232
Place/LocalBusiness/Restaurant
Seattle, WA, US - Downtown

Truck Stop Cafe
Place/LocalBusiness/Restaurant
Seattle, WA, US - Denny Triangle

Sogo Brazilian Grill - (206) 453-4366
Place/LocalBusiness/Restaurant
Seattle, WA, US - Downtown

NOTE

The JSON and HTTP headings at the bottom of the page reveal the JSON response and HTTP request information when clicked. These details are useful when exploring the service.

The tutorial app illustrates how to:

- Perform a Bing Entity Search API call in JavaScript
- Perform a Bing Maps `locationQuery` API call in JavaScript
- Pass search options to the API calls
- Display search results

- Handle the Bing client ID and API subscription keys
- Deal with any errors that might occur

The tutorial page is entirely self-contained; it does not use any external frameworks, style sheets, or even image files. It uses only widely supported JavaScript language features and works with current versions of all major Web browsers.

In this tutorial, we discuss only selected portions of the source code. The full source code is available on a separate page. Copy and paste this code into a text editor and save it as `bing.html`.

> **NOTE**
> This tutorial is substantially similar to the single-page Bing Web Search app tutorial, but deals only with entity search results.

App components

Like any single-page Web app, the tutorial application includes three parts:

- HTML - Defines the structure and content of the page
- CSS - Defines the appearance of the page
- JavaScript - Defines the behavior of the page

This tutorial doesn't cover most of the HTML or CSS in detail, as they are straightforward.

The HTML contains the search form in which the user enters a query and chooses search options. The form is connected to the JavaScript that actually performs the search by the `<form>` tag's `onsubmit` attribute:

```
<form name="bing" onsubmit="return newBingEntitySearch(this)">
```

The `onsubmit` handler returns `false`, which keeps the form from being submitted to a server. The JavaScript code actually does the work of collecting the necessary information from the form and performing the search.

The search is done in two phases. First, if the user has entered a location restriction, a Bing Maps query is done to convert it into coordinates. The callback for this query then kicks off the Bing Entity Search query.

The HTML also contains the divisions (HTML `<div>` tags) where the search results appear.

Managing subscription keys

> **NOTE**
> This app requires subscription keys for both the Bing Search API and the Bing Maps API. You can use a trial Bing Search key and a basic Bing Maps key.

To avoid having to include the Bing Search and Bing Maps API subscription keys in the code, we use the browser's persistent storage to store them. If either key has not been stored, we prompt for it and store it for later use. If the key is later rejected by the API, we invalidate the stored key so the user is asked for it upon their next search.

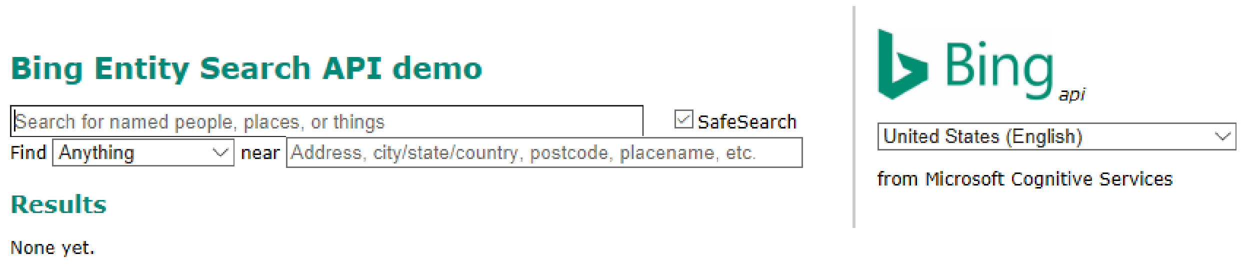
We define `storeValue` and `retrieveValue` functions that use either the `localStorage` object (if the browser supports it) or a cookie. Our `getSubscriptionKey()` function uses these functions to store and retrieve the user's key.

```
// cookie names for data we store
SEARCH_API_KEY_COOKIE = "bing-search-api-key";
MAPS_API_KEY_COOKIE   = "bing-maps-api-key";
CLIENT_ID_COOKIE      = "bing-search-client-id";

// API endpoints
SEARCH_ENDPOINT = "https://api.cognitive.microsoft.com/bing/v7.0/entities";
MAPS_ENDPOINT   = "http://dev.virtualearth.net/REST/v1/Locations";

// ... omitted definitions of storeValue() and retrieveValue()

// get stored API subscription key, or prompt if it's not found
function getSubscriptionKey(cookie_name, key_length, api_name) {
    var key = retrieveValue(cookie_name);
    while (key.length !== key_length) {
        key = prompt("Enter " + api_name + " API subscription key:", "").trim();
    }
    // always set the cookie in order to update the expiration date
    storeValue(cookie_name, key);
    return key;
}

function getMapsSubscriptionKey() {
    return getSubscriptionKey(MAPS_API_KEY_COOKIE, 64, "Bing Maps");
}

function getSearchSubscriptionKey() {
    return getSubscriptionKey(SEARCH_API_KEY_COOKIE, 32, "Bing Search");
}
```

The HTML `<body>` tag includes an `onload` attribute that calls `getSearchSubscriptionKey()` and `getMapsSubscriptionKey()` when the page has finished loading. These calls serve to immediately prompt the user for their keys if they haven't yet entered them.

```
<body onload="document.forms.bing.query.focus(); getSearchSubscriptionKey(); getMapsSubscriptionKey();">
```

Selecting search options


Bing Entity Search API demo
Search for named people, places, or things
SafeSearch
Find Anything
near Address, city/state/country, postcode, placename, etc.
Results
None yet.
Bing api
United States (English)
from Microsoft Cognitive Services

The HTML form includes the following controls:

`where`	A drop-down menu for selecting the market (location and language) used for the search.
`query`	The text field in which to enter the search terms.
`safe`	A checkbox indicating whether SafeSearch is turned on (restricts "adult" results)

`what`	A menu for choosing to search for entities, places, or both.
`mapquery`	The text field in which the user may enter a full or partial address, a landmark, etc. to help Bing Entity Search return more relevant results.

NOTE

Places results are currently available only in the United States. The `where` and `what` menus have code to enforce this restriction. If you choose a non-US market while Places is selected in the `what` menu, `what` changes to Anything. If you choose Places while a non-US market is selected in the `where` menu, `where` changes to the US.

Our JavaScript function `bingSearchOptions()` converts these fields to a partial query string for the Bing Search API.

```
// build query options from the HTML form
function bingSearchOptions(form) {

    var options = [];
    options.push("mkt=" + form.where.value);
    options.push("SafeSearch=" + (form.safe.checked ? "strict" : "off"));
    if (form.what.selectedIndex) options.push("responseFilter=" + form.what.value);
    return options.join("&");
}
```

For example, the SafeSearch feature can be `strict`, `moderate`, or `off`, with `moderate` being the default. But our form uses a checkbox, which has only two states. The JavaScript code converts this setting to either `strict` or `off` (we don't use `moderate`).

The `mapquery` field isn't handled in `bingSearchOptions()` because it is used for the Bing Maps location query, not for Bing Entity Search.

Obtaining a location

The Bing Maps API offers a `locationQuery` method, which we use to find the latitude and longitude of the location the user enters. These coordinates are then passed to the Bing Entity Search API with the user's request. The search results prioritize entities and places that are close to the specified location.

We can't access the Bing Maps API using an ordinary `XMLHttpRequest` query in a Web app because the service does not support cross-origin queries. Fortunately, it supports JSONP (the "P" is for "padded"). A JSONP response is an ordinary JSON response wrapped in a function call. The request is made by inserting using a `<script>` tag into the document. (Loading scripts is not subject to browser security policies.)

The `bingMapsLocate()` function creates and inserts the `<script>` tag for the query. The `jsonp=bingMapsCallback` segment of the query string specifies the name of the function to be called with the response.

```
function bingMapsLocate(where) {

    where = where.trim();
    var url = MAPS_ENDPOINT + "?q=" + encodeURIComponent(where) +
                "&jsonp=bingMapsCallback&maxResults=1&key=" + getMapsSubscriptionKey();

    var script = document.getElementById("bingMapsResult")
    if (script) script.parentElement.removeChild(script);

    // global variable holds reference to timer that will complete the search if the maps query fails
    timer = setTimeout(function() {
        timer = null;
        var form = document.forms.bing;
        bingEntitySearch(form.query.value, "", bingSearchOptions(form), getSearchSubscriptionKey());
    }, 5000);

    script = document.createElement("script");
    script.setAttribute("type", "text/javascript");
    script.setAttribute("id", "bingMapsResult");
    script.setAttribute("src", url);
    script.setAttribute("onerror", "BingMapsCallback(null)");
    document.body.appendChild(script);

    return false;
}
```

NOTE

If the Bing Maps API does not respond, the `bingMapsCallBack()` function is never called. Ordinarily, that would mean that `bingEntitySearch()` isn't called, and the entity search results do not appear. To avoid this scenario, `bingMapsLocate()` also sets a timer to call `bingEntitySearch()` after five seconds. There is logic in the callback function to avoid performing the entity search twice.

When the query completes, the `bingMapsCallback()` function is called, as requested.

```
function bingMapsCallback(response) {

    if (timer) {    // we beat the timer; stop it from firing
        clearTimeout(timer);
        timer = null;
    } else {        // the timer beat us; don't do anything
        return;
    }

    var location = "";
    var name = "";
    var radius = 1000;

    if (response) {
        try {
            if (response.statusCode === 401) {
                invalidateMapsKey();
            } else if (response.statusCode === 200) {
                var resource = response.resourceSets[0].resources[0];
                var coords   = resource.point.coordinates;
                name         = resource.name;

                // the radius is the largest of the distances between the location and the corners
                // of its bounding box (in case it's not in the center) with a minimum of 1 km
                try {
                    var bbox    = resource.bbox;
                    radius  = Math.max(haversineDistance(bbox[0], bbox[1], coords[0], coords[1]),
                                       haversineDistance(coords[0], coords[1], bbox[2], bbox[1]),
                                       haversineDistance(bbox[0], bbox[3], coords[0], coords[1]),
                                       haversineDistance(coords[0], coords[1], bbox[2], bbox[3]), 1000);
                } catch(e) {  }
                var location = "lat:" + coords[0] + ";long:" + coords[1] + ";re:" + Math.round(radius);
            }
        }
        catch (e) { }   // response is unexpected. this isn't fatal, so just don't provide location
    }

    var form = document.forms.bing;
    if (name) form.mapquery.value = name;
    bingEntitySearch(form.query.value, location, bingSearchOptions(form), getSearchSubscriptionKey());

}
```

Along with latitude and longitude, the Bing Entity Search query requires a *radius* that indicates the precision of the location information. We calculate the radius using the *bounding box* provided in the Bing Maps response. The bounding box is a rectangle that surrounds the entire location. For example, if the user enters `NYC`, the result contains roughly central coordinates of New York City and a bounding box that encompasses the city.

We first calculate the distances from the primary coordinates to each of the four corners of the bounding box using the function `haversineDistance()` (not shown). We use the largest of these four distances as the radius. The minimum radius is a kilometer. This value is also used as a default if no bounding box is provided in the response.

Having obtained the coordinates and the radius, we then call `bingEntitySearch()` to perform the actual search.

Performing the search

Given the query, a location, an options string, and the API key, the `BingEntitySearch()` function makes the Bing Entity Search request.

```
// perform a search given query, location, options string, and API keys
function bingEntitySearch(query, latlong, options, key) {

    // scroll to top of window
    window.scrollTo(0, 0);
    if (!query.trim().length) return false;     // empty query, do nothing

    showDiv("noresults", "Working. Please wait.");
    hideDivs("pole", "mainline", "sidebar", "_json", "_http", "error");

    var request = new XMLHttpRequest();
    var queryurl = SEARCH_ENDPOINT + "?q=" + encodeURIComponent(query) + "&" + options;

    // open the request
    try {
        request.open("GET", queryurl);
    }
    catch (e) {
        renderErrorMessage("Bad request (invalid URL)\n" + queryurl);
        return false;
    }

    // add request headers
    request.setRequestHeader("Ocp-Apim-Subscription-Key", key);
    request.setRequestHeader("Accept", "application/json");

    var clientid = retrieveValue(CLIENT_ID_COOKIE);
    if (clientid) request.setRequestHeader("X-MSEdge-ClientID", clientid);

    if (latlong) request.setRequestHeader("X-Search-Location", latlong);

    // event handler for successful response
    request.addEventListener("load", handleBingResponse);

    // event handler for erorrs
    request.addEventListener("error", function() {
        renderErrorMessage("Error completing request");
    });

    // event handler for aborted request
    request.addEventListener("abort", function() {
        renderErrorMessage("Request aborted");
    });

    // send the request
    request.send();
    return false;
}
```

Upon successful completion of the HTTP request, JavaScript calls our `load` event handler, the `handleBingResponse()` function, to handle a successful HTTP GET request to the API.

```
// handle Bing search request results
function handleBingResponse() {
    hideDivs("noresults");

    var json = this.responseText.trim();
    var jsobj = {};

    // try to parse JSON results
    try {
        if (json.length) jsobj = JSON.parse(json);
    } catch(e) {
        renderErrorMessage("Invalid JSON response");
    }

    // show raw JSON and HTTP request
    showDiv("json", preFormat(JSON.stringify(jsobj, null, 2)));
    showDiv("http", preFormat("GET " + this.responseURL + "\n\nStatus: " + this.status + " " +
        this.statusText + "\n" + this.getAllResponseHeaders()));

    // if HTTP response is 200 OK, try to render search results
    if (this.status === 200) {
        var clientid = this.getResponseHeader("X-MSEdge-ClientID");
        if (clientid) retrieveValue(CLIENT_ID_COOKIE, clientid);
        if (json.length) {
            if (jsobj._type === "SearchResponse") {
                renderSearchResults(jsobj);
            } else {
                renderErrorMessage("No search results in JSON response");
            }
        } else {
            renderErrorMessage("Empty response (are you sending too many requests too quickly?)");
        }
    if (divHidden("pole") && divHidden("mainline") && divHidden("sidebar"))
        showDiv("noresults", "No results.<p><small>Looking for restaurants or other local businesses? Those
currently areen't supported outside the US.</small>");
    }

    // Any other HTTP status is an error
    else {
        // 401 is unauthorized; force re-prompt for API key for next request
        if (this.status === 401) invalidateSearchKey();

        // some error responses don't have a top-level errors object, so gin one up
        var errors = jsobj.errors || [jsobj];
        var errmsg = [];

        // display HTTP status code
        errmsg.push("HTTP Status " + this.status + " " + this.statusText + "\n");

        // add all fields from all error responses
        for (var i = 0; i < errors.length; i++) {
            if (i) errmsg.push("\n");
            for (var k in errors[i]) errmsg.push(k + ": " + errors[i][k]);
        }

        // also display Bing Trace ID if it isn't blocked by CORS
        var traceid = this.getResponseHeader("BingAPIs-TraceId");
        if (traceid) errmsg.push("\nTrace ID " + traceid);

        // and display the error message
        renderErrorMessage(errmsg.join("\n"));
    }
}
```

IMPORTANT

A successful HTTP request does *not* necessarily mean that the search itself succeeded. If an error occurs in the search operation, the Bing Entity Search API returns a non-200 HTTP status code and includes error information in the JSON response. Additionally, if the request was rate-limited, the API returns an empty response.

Much of the code in both of the preceding functions is dedicated to error handling. Errors may occur at the following stages:

STAGE	POTENTIAL ERROR(S)	HANDLED BY
Building JavaScript request object	Invalid URL	`try` / `catch` block
Making the request	Network errors, aborted connections	`error` and `abort` event handlers
Performing the search	Invalid request, invalid JSON, rate limits	tests in `load` event handler

Errors are handled by calling `renderErrorMessage()` with any details known about the error. If the response passes the full gauntlet of error tests, we call `renderSearchResults()` to display the search results in the page.

Displaying search results

The Bing Entity Search API requires you to display results in a specified order. Since the API may return two different kinds of responses, it is not enough to iterate through the top-level `Entities` or `Places` collection in the JSON response and display those results. (If you want only one type of result, use the `responseFilter` query parameter.)

Instead, we use the `rankingResponse` collection in the search results to order the results for display. This object refers to items in the `Entitiess` and/or `Places` collections.

`rankingResponse` may contain up to three collections of search results, designated `pole`, `mainline`, and `sidebar`.

`pole`, if present, is the most relevant search result and should be displayed prominently. `mainline` refers to the bulk of the search results. Mainline results should be displayed immediately after `pole` (or first, if `pole` is not present).

Finally. `sidebar` refers to auxiliary search results. They may be displayed in an actual sidebar or simply after the mainline results. We have chosen the latter for our tutorial app.

Each item in a `rankingResponse` collection refers to the actual search result items in two different, but equivalent, ways.

`id`	The `id` looks like a URL, but should not be used for links. The `id` type of a ranking result matches the `id` of either a search result item in an answer collection, *or* an entire answer collection (such as `Entities`).
`answerType` `resultIndex`	The `answerType` refers to the top-level answer collection that contains the result (for example, `Entities`). The `resultIndex` refers to the result's index within that collection. If `resultIndex` is omitted, the ranking result refers to the entire collection.

NOTE

For more information on this part of the search response, see Rank Results.

You may use whichever method of locating the referenced search result item is most convenient for your application. In our tutorial code, we use the `answerType` and `resultIndex` to locate each search result.

Finally, it's time to look at our function `renderSearchResults()`. This function iterates over the three `rankingResponse` collections that represent the three sections of the search results. For each section, we call `renderResultsItems()` to render the results for that section.

```
// render the search results given the parsed JSON response
function renderSearchResults(results) {

    // if spelling was corrected, update search field
    if (results.queryContext.alteredQuery)
        document.forms.bing.query.value = results.queryContext.alteredQuery;

    // for each possible section, render the resuts from that section
    for (section in {pole: 0, mainline: 0, sidebar: 0}) {
        if (results.rankingResponse[section])
            showDiv(section, renderResultsItems(section, results));
    }
}
```

Rendering result items

In our JavaScript code is an object, `searchItemRenderers`, that contains *renderers:* functions that generate HTML for each kind of search result.

```
searchItemRenderers = {
    entities: function(item) { ... },
    places: function(item) { ... }
}
```

A renderer function may accept the following parameters:

`item`	The JavaScript object containing the item's properties, such as its URL and its description.
`index`	The index of the result item within its collection.
`count`	The number of items in the search result item's collection.

The `index` and `count` parameters can be used to number results, to generate special HTML for the beginning or end of a collection, to insert line breaks after a certain number of items, and so on. If a renderer does not need this functionality, it does not need to accept these two parameters. In fact, we do not use them in the renderers for our tutorial app.

Let's take a closer look at the `entities` renderer:

```
    entities: function(item) {
        var html = [];
        html.push("<p class='entity'>");
        if (item.image) {
            var img = item.image;
            if (img.hostPageUrl) html.push("<a href='" + img.hostPageUrl + "'>");
            html.push("<img src='" + img.thumbnailUrl +  "' title='" + img.name + "' height=" + img.height +
" width= " + img.width + ">");
            if (img.hostPageUrl) html.push("</a>");
            if (img.provider) {
                var provider = img.provider[0];
                html.push("<small>Image from ");
                if (provider.url) html.push("<a href='" + provider.url + "'>");
                html.push(provider.name ? provider.name : getHost(provider.url));
                if (provider.url) html.push("</a>");
                html.push("</small>");
            }
        }
        html.push("<p>");
        if (item.entityPresentationInfo) {
            var pi = item.entityPresentationInfo;
            if (pi.entityTypeHints || pi.entityTypeDisplayHint) {
                html.push("<i>");
                if (pi.entityTypeDisplayHint) html.push(pi.entityTypeDisplayHint);
                else if (pi.entityTypeHints) html.push(pi.entityTypeHints.join("/"));
                html.push("</i> - ");
            }
        }
        html.push(item.description);
        if (item.webSearchUrl) html.push("&nbsp;<a href='" + item.webSearchUrl + "'>More</a>")
        if (item.contractualRules) {
            html.push("<p><small>");
            var rules = [];
            for (var i = 0; i < item.contractualRules.length; i++) {
                var rule = item.contractualRules[i];
                var link = [];
                if (rule.license) rule = rule.license;
                if (rule.url) link.push("<a href='" + rule.url + "'>");
                link.push(rule.name || rule.text || rule.targetPropertyName + " source");
                if (rule.url) link.push("</a>");
                rules.push(link.join(""));
            }
            html.push("License: " + rules.join(" - "));
            html.push("</small>");
        }
        return html.join("");
    }, // places renderer omitted
```

Our entity renderer function:

- Builds the HTML `<img>` tag to display the image thumbnail, if any.
- Builds the HTML `<a>` tag that links to the page that contains the image.
- Builds the description that displays information about the image and the site it's on.
- Incorporates the entity's classification using the display hints, if any.
- Includes a link to a Bing search to get more information about the entity.
- Displays any licensing or attribution information required by data sources.

Persisting client ID

Responses from the Bing search APIs may include a `X-MSEdge-ClientID` header that should be sent back to the API with successive requests. If multiple Bing Search APIs are being used, the same client ID should be used with all of them, if possible.

Providing the `X-MSEdge-ClientID` header allows the Bing APIs to associate all of a user's searches, which has two important benefits.

First, it allows the Bing search engine to apply past context to searches to find results that better satisfy the user. If a user has previously searched for terms related to sailing, for example, a later search for "docks" might preferentially return information about places to dock a sailboat.

Second, Bing may randomly select users to experience new features before they are made widely available. Providing the same client ID with each request ensures that users that have been chosen to see a feature always see it. Without the client ID, the user might see a feature appear and disappear, seemingly at random, in their search results.

Browser security policies (CORS) may prevent the `X-MSEdge-ClientID` header from being available to JavaScript. This limitation occurs when the search response has a different origin from the page that requested it. In a production environment, you should address this policy by hosting a server-side script that does the API call on the same domain as the Web page. Since the script has the same origin as the Web page, the `X-MSEdge-ClientID` header is then available to JavaScript.

NOTE

In a production Web application, you should perform the request server-side anyway. Otherwise, your Bing Search API key must be included in the Web page, where it is available to anyone who views source. You are billed for all usage under your API subscription key, even requests made by unauthorized parties, so it is important not to expose your key.

For development purposes, you can make the Bing Web Search API request through a CORS proxy. The response from such a proxy has an `Access-Control-Expose-Headers` header that whitelists response headers and makes them available to JavaScript.

It's easy to install a CORS proxy to allow our tutorial app to access the client ID header. First, if you don't already have it, install Node.js. Then issue the following command in a command window:

```
npm install -g cors-proxy-server
```

Next, change the Bing Web Search endpoint in the HTML file to:

```
http://localhost:9090/https://api.cognitive.microsoft.com/bing/v7.0/search
```

Finally, start the CORS proxy with the following command:

```
cors-proxy-server
```

Leave the command window open while you use the tutorial app; closing the window stops the proxy. In the expandable HTTP Headers section below the search results, you can now see the `X-MSEdge-ClientID` header (among others) and verify that it is the same for each request.

Next steps

Bing Entity Search API reference

Bing Maps API documentation