



NAICS Association Business Intelligence API

Version 2.0.7.1
 Copyright © 2017-2018
 Updated 12/18/18

Table of Contents

0.0 [Test the BizAPI Service without coding](#)

- 1. Making a Request
 - 1.1 POST
 - 1.2 Response
 - 1.3 Record Types

Search Terms (Input Data) –	Included with each Response
Match Data –	Included with each Response
NAICS & SIC Telemarketing Enhanced Telemarketing with Employee Figures Enhanced Telemarketing with Sales Figures Prospecting	

1.4 Input Data Quality

- 2. Understanding Match Data
 - 2.1 BEMFAB
 - 2.2 Match Grade
 - 2.3 Confidence Code
 - 2.4 DUNS #
 - 2.5 Matches Remaining

Sample Code Snippets in:

PHP	ASP	JavaScript with jQuery
---------------------	---------------------	--

3. Data Flow Architecture

0.0 Test the BizAPI without coding

You can test the API service without doing any coding. Just go to this link, <https://www.naics.com/naics-api-test/>, enter your credentials and fill the form with the Company record data. When submitted, you will see the JSON result of your request.

1.0 Making a Request

The API is organized around REST. All requests should be made over SSL. All request and response bodies, including errors, will be encoded in JSON. Requests are Authenticated via Basic Authentication.

The BizAPI Utilizes Basic Authentication. Credentials are given to you during your Account Activation.

For Example:

Username: JohnDoeInc

Password: xxxx xxxx xxxx xxxx xxxx xxxx

1.1 POST

POST <https://www.naics.com/wp-json/naicsapi/v1/cosearch>

The POST body should contain the following fields:

```
{
  "companyName": "<company name>",
  "state": "<state>",
  "country": "<country>",
  "address": "<address>",
  "city": "<city>",
  "phone": "<phone>",
  "postalCode": "<post code>",
}
```

You can See Sample Code in [ASP](#), [PHP](#), and [JavaScript with jQuery](#) at the bottom of this Document.

1.2 Response

The BizAPI returns standard Success and Error Codes.

HTTP Status codes

200 OK

400 "[companyName](#), [city](#), [state](#) and [country](#) are required fields. "

401 Credentials Missing or Invalid.

403 "[Request not submitted due to lack of searches. Contact \[apicloudsolutions@naics.com\]\(mailto:apicloudsolutions@naics.com\) or call 973-625- 5626 to purchase more searches.](#) "

If a match can be found, the following JSON will be returned, based on the record layout Established when Credentials were created. See [Record Types](#) for more information.

Below is an example of all possible fields.

```
{
  "Search Terms": {
    "Company Name": "...",
    "Street Address": "...",
    "City": "...",
    "State": "...",
    "Zip Code": "...",
    "Country": "...",
    "Phone": "..."
  },
  "Matching Data": {
    "BEMFAB": "...",
    "Match Grade": "...",
    "Confidence Code": "...",
    "DUNS #": "...",
    "Matches Remaining": integer,
  },
  "Appended Data": {
    "Company Name": "...",
    "Secondary Business Name":
    "",
    "Address Source": "...",
    "Street Address": "...",
    "PO Box": "...",
    "State/Province": "...",
    "City": "...",
    "ZIP Code": "...",
    "Phone": "...",
    "CEO Title": "...",
    "CEO Name": "...",
    "CEO First Name": "...",
    "CEO Last Name": "...",
    "Line of Business": "...",
    "Location Type": "...",
    "Year Started": "...",
    "Employees on Site": "...",
    "Employees Total": "...",
    "Sales Volume": "...",
    "4 Digit SIC 1": "...",
    "4 Digit SIC 1 Description": "...",
    "4 Digit SIC 2": "...",
    "4 Digit SIC 2 Description": "...",
    "8 Digit SIC 1": "...",
    "8 Digit SIC 1 Description": "...",
    "8 Digit SIC 2": "...",
    "8 Digit SIC 2 Description": "...",
    "NAICS 1 Code": "...",
    "NAICS 1 Description": "...",
    "NAICS 2 Code": "...",
    "NAICS 2 Description": "..."
  }
}
```

NOTES:

Search Terms included in all Layouts

Matching Data included in all Layouts

- Only Data with 7 or greater Confidence Code Returned
- DUNS # will return like "xx-xxx-xxxx"
- Indicates How Many Search Credits Remain

Appended Data Returned Dependent on Layout Chosen

- Indicates whether Physical or Mailing Address Appended

- Indicates if Single Location, Headquarters, or Branch

- Indicates Employees on Location
- Indicates All Employees on or Off Location

- Reflects Official 4 Digit SIC code

- Reflects 4 Digit SIC code with D&B's proprietary 4 digit extension

- Reflects Official 6 Digit NAICS code

1.3 Record Types

Each Record type has a unique price tier. Contact APICloudSolutions@NAICS.com to acquire Pricing.

Each Response will include the [Search Terms \(Input Data\)](#), [Matching Data](#), and [Appended Data](#). The [Appended Data](#) Returned will be based on one of the following Layouts:

NAICS & SIC Layout

Includes the Following:

4 Digit SIC 1	8 Digit SIC 2
4 Digit SIC 1 Description	8 Digit SIC 2 Description
4 Digit SIC 2	NAICS 1 Code
4 Digit SIC 2 Description	NAICS 1 Description
8 Digit SIC 1	NAICS 2 Code
8 Digit SIC 1 Description	NAICS 2 Description

Telemarketing Layout

Includes [NAICS & SIC](#) content PLUS:

Company Name	Phone
Secondary Business Name	CEO Title
Address Source	CEO Name
Street Address	CEO First Name
PO Box	CEO Last Name
State/Province	Line of Business
City	Location Type
ZIP Code	Year Started

Enhanced Telemarketing with Employee Figures Layout

Includes [NAICS & SIC](#) + [Telemarketing](#) content PLUS:

Employees on Site
Employees Total

Enhanced Telemarketing with Sales Figures Layout

Includes [NAICS & SIC](#) + [Telemarketing](#) content PLUS:

Sales Volume

Prospecting Layout

Includes [NAICS & SIC](#) + [Telemarketing](#) content PLUS:

Employees on Site
Employees Total
Sales Volume

1.4 Improving Input Data Quality

The User should get data for their match project from the highest and best source of data they have internally. (Often, highest ≠ best)

- By Highest, we mean the source that is closest to the point of data entry (when they open a new account, the data is entered at that point.)
- By Best, we mean the source that has the most accurate data contained in the match input fields and it should have the most complete data (see input layout) available.

Other factors, which will affect input file accuracy, include situations where content is contained in the wrong field. For instance, we frequently see situations where the User will contain information in their Address field, which in fact is not the customer's address.

We are listing here some common examples of this:

1. Where the address field contains information other than an address. For instance:
 - a) A person's name or
 - b) ATTN TO:
 - c) A Department Name
 - d) A second Company Name
2. Where the address populated is actually not the address of the Company Name provided. This frequently occurs where the User sells through dealers and/or Value Added Resellers (VARs).

Where possible, the User should eliminate or correct the bias these inaccuracies cause before submitting a request.

Request Parameters

The BizAPI utilizes intuitive Matching software to match against Records written in many different styles.

Below are some insights to ensure you are maximizing the opportunities to Append successfully:

- Each Field can handle over 200 characters. 65 characters or less is generally better for matching.
- No layout needs to be specified when submitting a request. Layout is established upon API Credential Activation.
- Special Characters such as Ò ¸ œ and © will not be acknowledged by the API and may be turned into a question mark (?) if a match is found. Avoid using special characters if you can.
- "**address**": "<address>" Should only Contain the Street Address OR a PO Box #.
- "**phone**": "<phone>" can handle most Standard Phone Formats. Don't include Extensions. Use a string of numbers with no Special characters to ensure the highest chance for a match.
- Use Data written in a Standard Postal Code convention when able.
Ex. Avenue can be written as: Av, Ave, Aven, Avenu, Avenue, Avn, Avnue.

Completeness

The customer should supply as many of the fields on the input layout as possible. Not only should they supply as many of the fields, but also the fields should be populated as completely as possible. Only Records matched at a Confidence score of 7 or higher will be appended, so Records that lack 3 or more fields are not likely to match at a sufficient Confidence Level to Append.

2.0 Understanding Matching Data

Matching Data is included to help you understand the Quality of the Appended data Matched to your Input Data.

There are 5 Fields included:

"BEMFAB":

"Match Grade":

"Confidence Code":

"DUNS #":

"Matches Remaining":

2.1 Bemfab

The Bemfab indicator gives insight about the Marketability of the Appended data. It will return a Single Letter Response or return as blank. These are the meanings behind each Possible Return Value:

Blank: Marketability is undefined.

M = Matched to Marketable Record File.

A = Dun's numbered record that has been flagged as undeliverable. Industry will be appended.

D = At the customer's request the record has been De-listed. Can't sell company name but can append Industry code.

O = Unconfirmed record or confirmed as Out of business.

2.2 Match Grade

The Match Grade will return a 7 Letter String when a Successful Match occurs. Each letter indicates how successfully a piece of the Input Data was matched to the Database Record.

Here is an example of how the String might look: ABAAAFZ

This is the sequence in which the Match Grade string reflects each data element.

1. Company Name
2. Street Address Number
3. Street Name
4. City
5. State
6. PO Box*
7. Telephone

Letter Grades should be interpreted in the following way:

A - Same: ABC WIDGET MFG vs. ABC Widgeting MFG

B - Similar: ABC Widget MFG vs. ABC MFG

F - Different: ABC Widget MFG vs. XYZ MFG

Z - Null: One or Both are Null (Blank)

*Note: When you submit a record, you can supply Either a Street Address or a PO Box in the "[address](#)": "<[address](#)>" field. If the Record fails to Match on Street Address, it is recommended you try again using the PO Box Address.

2.3 Confidence Code

Confidence codes indicate the strength of match or our "Confidence Level" in that particular match. 10 is the highest confidence level for a matched record down to 4 which is the loosest match. The API only appends Records with a Confidence Level of 7 or Higher. Typically all records matched at 7 or higher are strong matches. 6's can be strong matches, but not consistently enough to allow through the API. 5's and 4's are either just a Company Name or Physical Address match so we typically don't append to them.

2.4 DUNS

The D&B D-U-N-S® Number is a unique nine-digit identifier for businesses. It is used to establish a business credit file, which is often referenced by lenders and potential business partners to help predict the reliability and/or financial stability of the company in question. D-U-N-S, which stands for data universal number system, is used to track and maintain accurate and timely information on +265M global businesses. (source- <http://www.dnb.com/duns-number.html>)

2.5 Matches Remaining

This Field indicates how Many Search Credits you have remaining before the API will stop working. To Increase your Credits, or request assistance please email us at APICloudSolutions@NAICS.com, or call 973-625-5626.

CODE SNIPPET FOR PHP

```
<?php
```

```
$username = "xxx"; //Replace xxx with your username.
$password = "xxx"; //Replace xxx with yor password.

$credentials = base64_encode($username." ".$password);

if (!empty($_POST)){
    // Collect data posted from form into an array.
    $companyInfo = array(
        'companyName' => $_POST['companyName'],
        'address' => $_POST['address'],
        'city' => $_POST['city'],
        'state' => $_POST['state'],
        'country' => $_POST['country'],
        'phone' => $_POST['phone'],
        'postalCode' => $_POST['postalCode'],
    );

    // Prepare REST request
    // Get cURL resource
    $ch = curl_init();

    // Set url
    curl_setopt($ch, CURLOPT_URL, https://www.naics.com/wp-json/naicsapi/v1/cosearch/);
    // Set method
    curl_setopt($ch, CURLOPT_CUSTOMREQUEST, 'POST');
    // Set options
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
    // Set headers
    curl_setopt($ch, CURLOPT_HTTPHEADER, [
        "Authorization: Basic ".$credentials,
        "Content-Type: application/json; charset=utf-8",
    ]
    );

    // Create body
    $body = json_encode($companyInfo);

    // Set body
    curl_setopt($ch, CURLOPT_POST, 1);
    curl_setopt($ch, CURLOPT_POSTFIELDS, $body);

    // Send the request & save response to $response
    $response = curl_exec($ch);

    if(!$response) {
        die('Error: "' . curl_error($ch) . "' - Code: ' . curl_errno($ch));
    } else {
        $status = "Response HTTP Status Code : " . curl_getinfo($ch, CURLINFO_HTTP_CODE);
        // Convert JSON reply to a PHP array
    }
}
```



```

    $respArray = json_decode($response, true);
    // Build HTML output
    $htmlContent = "<ul>";
    foreach ($respArray as $section=>$sectionArray){
        $htmlContent .= "<li><h4>";
        $htmlContent .= $section;
        $htmlContent .= "</h4>";
        $htmlContent .= "<ul>";
        foreach ($respArray[$section] as $key=>$value){
            $htmlContent .= "<li>";
            $htmlContent .= $key . ": <strong>". $value . "</strong>";
            $htmlContent .= "</li>";
        }
        $htmlContent .= "</ul></li>";
    }
    $htmlContent .= "</ul>";
}
// Close request to clear up some resources
curl_close($ch);
}
?>
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <title>NAICS Association API Sample Code</title>
    </head>
    <body>
        <h1>NAICS Association API Sample Page</h1>
        <form method="post">
            Company name*:<br>
            <input type="text" name="companyName" required><br>
            Address:<br>
            <input type="text" name="address"><br>
            City:<br>
            <input type="text" name="city"><br>
            State*:<br>
            <input type="text" name="state" required><br>
            Country*:<br>
            <input type="text" name="country" required><br>
            Phone:<br>
            <input type="text" name="phone"><br>
            Postal Code:<br>
            <input type="text" name="postalCode"><br>
            <input type="submit" value="Submit">
        </form>
        <p><?php echo $status; ?></p>
        <?php echo $htmlContent; ?>
    </body>
</html>

```

CODE SNIPPET FOR ASP

```
<%
if Request.Form("go")=1 then
  username=Request.Form("User")
  password=Request.Form("pwd")

end if
%>
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>NAICS Association API Sample Code</title>
    <script src="https://code.jquery.com/jquery-3.2.1.min.js" integrity="sha256-
hwg4gsxgFZhOsEEamdOYGBf13FyQuiTwIAQgxVSNg4=" crossorigin="anonymous"></script>
    <script type="text/javascript">
      $(document).ready(function(){
        $('#searchForm').submit(function(e){
          e.preventDefault();

          // The username and password below will be saved in clear text and available
          // to anybody who has access to the source code of this page.
          // This sample code is for testing and demonstration purposes only. Passwords
          // and usernames should never be stored this way on
          // a page that can be accessed in any way by users who are not trusted.
          var username = "<%=username%>"; //replace with your Username
          var password = "<%=password%>"; //replace with your password
          var form = $(this);
          searchFormData = form.serializeArray();
          var htmlContent = "";
          // send ajax
          jQuery.ajax({
            url: "https://www.naics.com/wp-json/naicsapi/v1/cosearch",
            type: "POST",
            headers: {
              "Authorization": "Basic " + btoa(username + ":" + password),
              "Content-Type": "application/json; charset=utf-8",
            },
            contentType: "application/json",
            data: JSON.stringify(convertFormData(searchFormData))
          })
          .done(function(naicsResponse, textStatus, jqXHR) {
            $('#requestStatus').html("HTTP Request Succeeded: " + jqXHR.status);
            $.each(naicsResponse, function(section, items){
              htmlContent += "<li><h4>";
              htmlContent += section;
              htmlContent += "</h4>";
              htmlContent += "<ul>";
              $.each (items, function(key, item){
                htmlContent += "<li>";
```

```

        htmlContent += key + ": <strong>" + item +
"</strong>";
        htmlContent += "</li>";
    });
    htmlContent += "</ul></li>";
    });
    $("#responseList").html(htmlContent);
}
.fail(function(jqXHR, textStatus, errorThrown) {
    $("#requestStatus").html("HTTP Request Failed: " + jqXHR.status);
    var failResponse = JSON.parse(jqXHR.responseText);
    $.each(failResponse, function(key, item){
        if (key != "data"){
            htmlContent += "<li>";
            htmlContent += key + ": <strong>" + item +
"</strong>";
            htmlContent += "</li>";
        }
    });
    $("#responseList").html(htmlContent);
})
.always(function() {
    // Clear form entries and reset default values after everything else
    has completed.
        $("#searchForm").trigger("reset");
    });
});
// Put form data into array that can converted into JSON
function convertFormData(data) {
    var unindexed_array = data;
    var indexed_array = {};
    $.map(unindexed_array, function(n, i) {
        indexed_array[n['name']] = n['value'];
    });
    return indexed_array;
}
</script>
</head>
<body>
    <h1>NAICS Association API Sample Page</h1>
<%
if Request.Form("go")<>"1" then
Response.Write("<form name=Credentials action=""ajaxAPI.asp"" method=post><input type=hidden name=go value=""1"">")
Response.Write("<table><tr><td align=right>User:</td><td><input type=password name=User size=50></td></tr>")
Response.Write("<table><tr><td align=right>Password:</td><td><input type=password name=pwd size=50></td></tr>")
Response.Write("<table><tr><td align=right></td><td><input type=submit value=""Go""></td></tr></table></form>")
else
%>
    <form id="searchForm" method="post">
        Company name*:<br>

```

```
<input type="text" name="companyName" required><br>
Address:<br>
<input type='text' name='address'><br>
City:<br>
<input type='text' name='city'><br>
State*:<br>
<input type='text' name='state' required><br>
Country*:<br>
<input type='text' name='country' required><br>
Phone:<br>
<input type='text' name='phone'><br>
Postal Code:<br>
<input type='text' name='postalCode'><br>
<input id="submitBtn" type='submit' value='Submit'>
</form>
<%
end if
%>
<p id="requestStatus"></p>
<ul id="responseList"></ul>
</body>
</html>
```

CODE SNIPPET FOR JavaScript with jQuery

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>NAICS Association API Sample Code</title>
    <script src="https://code.jquery.com/jquery-3.2.1.min.js" integrity="sha256-
hwg4gsgxFZhOsEEamdOYGBf13FyQuiTwIAQgxVSNgt4=" crossorigin="anonymous"></script>
    <script type="text/javascript">
      $(document).ready(function(){
        $("#searchForm").submit(function(e){
          e.preventDefault();

          // The username and password below will be saved in clear text and available
          // at anybody who has access to the source code of this page.
          // This sample code is for testing and demonstration purposes only. Passwords
          // and usernames should never be stored this way on
          // a page that can be accessed in any way by users who are not trusted.
          var username = "xxx"; // Replace xxx with your username.
          var password = "xxx"; // Replace xxx with your password.
          var form = $(this);
          searchFormData = form.serializeArray();
          var htmlContent = "";
          // send ajax
          jQuery.ajax({
            url: "https://www.naics.com/wp-json/naicsapi/v1/cosearch",
            type: "POST",
            headers: {
              "Authorization": "Basic " + btoa(username + ":" + password),
              "Content-Type": "application/json; charset=utf-8",
            },
            contentType: "application/json",
            data: JSON.stringify(convertFormData(searchFormData))
          })
          .done(function(naicsResponse, textStatus, jqXHR) {
            $( "#requestStatus" ).html("HTTP Request Succeeded: " + jqXHR.status);
            $.each(naicsResponse, function(section, items){
              htmlContent += "<li><h4>";
              htmlContent += section;
              htmlContent += "</h4>";
              htmlContent += "<ul>";
              $.each (items, function(key, item){
                htmlContent += "<li>";
                htmlContent += key + ": <strong>" + item +
                htmlContent += "</li>";
              });
              htmlContent += "</ul></li>";
            });
            $( "#responseList" ).html(htmlContent);
          })
          .fail(function(jqXHR, textStatus, errorThrown) {
            $( "#requestStatus" ).html("HTTP Request Failed: " + jqXHR.status);
            var failResponse = JSON.parse(jqXHR.responseText);
          });
        });
      });
    </script>
  </head>
  <body>
    <div id="searchForm">
      <input type="text" value="Search" />
      <input type="submit" value="Search" />
    </div>
    <div id="requestStatus">
      <strong>
        </strong>
    </div>
    <div id="responseList">
      <ul>
        </ul>
    </div>
  </body>
</html>
```

```

        $.each(failResponse, function(key, item){
            if (key != "data"){
                htmlContent += "<li>";
                htmlContent += key + ": <strong>" + item +

                htmlContent += "</li>";
            }
        });
        $("#responseList").html(htmlContent);
    })
    .always(function() {
        // Clear form entries and reset default values after everything else
        // has completed.

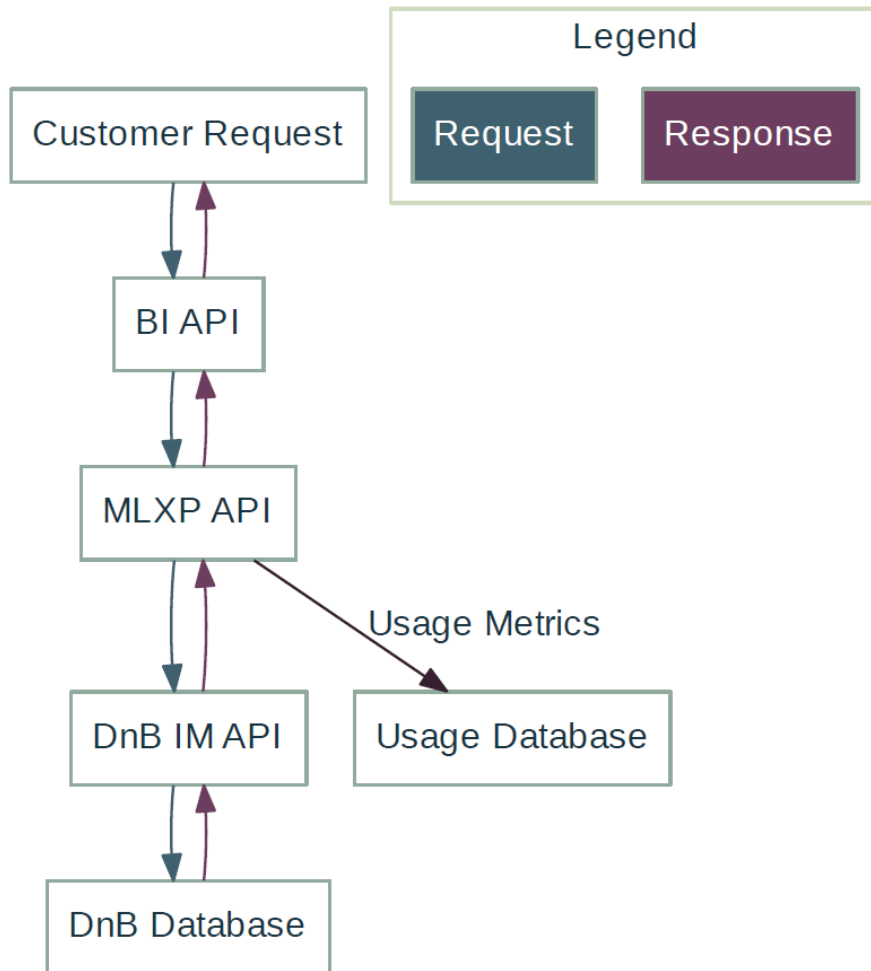
        $('#searchForm').trigger("reset");
    });
});

// Put form data into array that can converted into JSON
function convertFormData(data) {
    var unindexed_array = data;
    var indexed_array = {};
    $.map(unindexed_array, function(n, i) {
        indexed_array[n['name']] = n['value'];
    });
    return indexed_array;
}
</script>
</head>
<body>
<h1>NAICS Association API Sample Page</h1>
<form id="searchForm" method="post">
    Company name*:<br>
    <input type="text" name="companyName" required><br>
    Address:<br>
    <input type='text' name='address'><br>
    City:<br>
    <input type='text' name='city'><br>
    State*:<br>
    <input type='text' name='state' required><br>
    Country*:<br>
    <input type='text' name='country' required><br>
    Phone:<br>
    <input type='text' name='phone'><br>
    Postal Code:<br>
    <input type='text' name='postalCode'><br>
    <input id="submitBtn" type='submit' value='Submit'>
</form>
<p id="requestStatus"></p>
<ul id="responseList"></ul>
</body>
</html>

```

3.0 Data Flow Architecture

The following Architecture demonstrates the flow of Data from start to finish of the Request. Your input data is not stored on any server involved, at any point during the process.



NAICS Data Flow
Last Updated 2017-05-09