NAICS Association Business Intelligence API

Version 2.0.0.1

Updated 09/01/2021

Table of Contents

0.0 What Does the BizAPI Do?

The BizAPI is a real time solution that allows users to gain useful insights about a Business. The User submits one of a combination of Identifiable Information Fields about a Business (Company Name, Full Address, Phone; Company Name, Partial Address; Company Name/Country; URL; DUNS; Phone) and they are Matched against the D&B Business Database; the Largest and Best Quality Business Database in the Market. If a successful match is achieved, the BizAPI returns one of several Record Layouts, depending on which was Established when the API Account was created.

0.1 What Data Elements can the BizAPI Return?

What the BizAPI returns is dependent on which Record Layout is set up for your account. Available fields include:

- NAICS & SIC Codes
- Company Name & Address information as listed in the D&B Database. Useful for verifying the quality of the match made.
- Year Started, Location Type (Single Location, Headquarters, Branch)
- Company Size Figures (Number of Employees and Annual Sales) and URLs
- Corporate Linkage (who the Headquarter or Parent Company the Appended record reports to is; the Domestic Ultimate, and Global Ultimate Companies in the Family Tree of the Appended record; how many Establishments are in the same Family tree)

Note: Year Started, Total Employees and Annual Sales information will be blank for Branch locations. This information is only reported at the Headquarter Location to prevent confusion and the doubling of stats.

You can see the full breakout by Layout in Section 1.4 Record Types.

0.2 How Does Data Matching Work?

When submitting a request, choose one of the following Match methods based on the information you have available. The Match Methods are shown below in order from Most to Least Precise:

1. DUNS Match*
2. Standard Match (Company Name, Street/PO Box, City, State, Zip, Country, & Phone if available)
3. Loose Match (Company Name, partial address)
4. URL Match* (works for US Records only)
5. Name Match (Company Name & Country)
6. Phone Match*

*If any other fields are submitted with DUNS, URL, or Phone Match, then these match methods will not trigger.

Some users might benefit from submitting a request and, if it fails to append, resubmitting based on one of the other Match Methods available. For example, if you submit a DUNS number and no match is made, you can submit the Standard Match and likely find the correct DUNS number and record for the Company in your record.

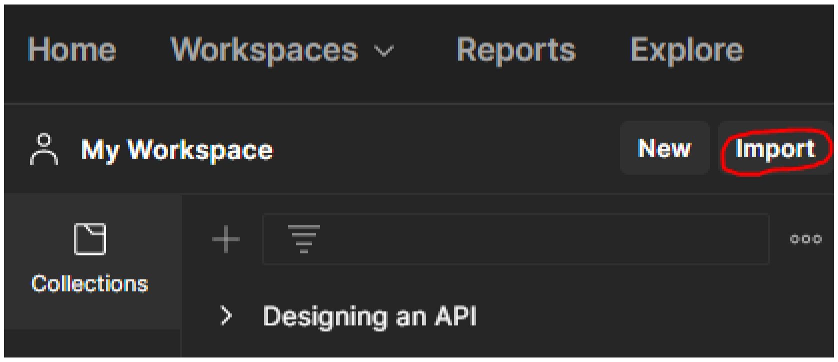
Records that Match based on DUNS, URL, or Phone will return a Confidence Level of 10 (Exact Match). Standard Matches will return a Confidence Level of 7 or higher, depending on how well each Data element matches; no Appended information is returned if a Confidence Level of 6 or lower is achieved since these almost always reflect bad results. Loose Matches and Name Matches will append at a Confidence Level of 8.

0.3 Testing the BizAPI Service prior to coding

You can test the API service without doing any coding. Just go to this link, https://www.naics.com/naics-api-test-V2/, enter your credentials and fill the form with the Company record data. When submitted, you will see the JSON result of your request.

0.4 Importing the Postman Collection

We put together some premade examples for use in Postman (https://www.postman.com/). Just download the JSON file at https://www.naics.com/bizapi-documents-v2/ and import it to Postman.



Then save your Basic Authentication Credentials to each premade example you want to interact with.

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. A maximum of 3 Requests can be made per rolling second. If you need this temporarily boosted for a large project, please reach out to your API Account contact or APICloudSolutions@NAICS.com for support.

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`
`Spaces in Password MUST be kept`

1.1 POST

POST https://naics.com/wp-json/naicsapi/v2/cosearch

Below are multiple representations of the POST body you can submit to achieve results (replace Fields in RED with Company Data.). You can review the Match Methods outlined in section 0.2 How does the Matching work?:

DUNS Match (will not trigger if other fields are present in request)

```
{
    "duns": "10-223-5004"
}
```

Standard Match

```
{
    "companyName": "Westrock Mwv, LLC",
    "address": "501 S 5TH St",
    "city": "Richmond",
    "state": "VA",
    "postalCode": "23219",
    "country": "US",
    "phone": "8044441000"
}
```

Loose Match (you can opt to include or exclude empty fields in your request). Requires at least Company Name and State.

```
{
    "companyName": "Westrock Mwv, LLC",
    "address": "",
    "city": "",
    "state": "VA",
    "postalCode": "",
    "country": "US",
    "phone": "8044441000"
}
```

URL Match (will not trigger if other fields are present in request) (works for US Records only)

```
{
    "url": "www.westrock.com"
}
```

Name Match (No Country Needed if US Location)

```
{
    "companyName": "Westrock Mwv, LLC"
}
```

Phone Match (will not trigger if other fields are present in request)

```
{
    "phone": "8044441000"
}
```

Note: You may include additional fields (excluding Personally Identifiable information) not represented above for your own reference. Inclusion of one of these additional fields will not prevent DUNS, URL, or Phone match from occurring. For example: You could add `"Client#": " 12345"` to the request and have it returned so you can better identify to which record to connect the result.

1.2 Response

The BizAPI returns standard Success and Error Codes.

HTTP Status codes

200	OK
400	'Bad Request', 'No valid search terms submitted. Must have at least one of "companyName", "duns", "url", or "phone".'
400	'Bad Request', 'Missing field: layout' (if this occurs, tell your NAICS API Contact)
401	Credentials are Missing or Invalid.
403	'Request not submitted due to lack of searches. Contact apicloudsolutions@naics.com or call 973-625-5626 to purchase more searches.'
429	'Too many requests. Please limit your requests to 3 per second'
500	Internal Server Error

If submission successfully processes (code 200) but no match can be found, the following JSON will be returned. `Text in Search Terms will reflect Company information submitted`

```
{
    "Search Terms": {
        "companyName": "Westrock Mwv, LLC",
        "address": "501 S 5TH St",
        "city": "Richmond",
        "state": "VA",
        "postalCode": "23219",
        "country": "US",
        "duns": "102235004",
        "url": "www.westrock.com",
        "phone": "8044441000"
    },
```

```json
    "Matching Data": {
        "Request ID": 471,
        "Layout": "PL",
        "Matches Remaining": 224344,
        "Match Method": ""
    },
    "Appended Data": {
        "Message": "No match found"
    }
}
```

If a match can be found, the following JSON will be returned, based on the record layout Established when Credentials were created. See Section 1.4 Record Layouts for more information. Below is an example of all possible fields (results continue onto next page).

```json
{
    "Search Terms": {
        "companyName": "Westrock Mwv, LLC",
        "address": "501 S 5TH St",
        "city": "Richmond",
        "state": "VA",
        "postalCode": "23219",
        "country": "US",
        "duns": "102235004",
        "url": "www.westrock.com",
        "phone": "8044441000"
    },
    "Matching Data": {
        "Request ID": 472,
        "Layout": "PL",
        "Matches Remaining": 224344,
        "Match Method": "Standard Match",
        "Match Grade": "AAAAAZA",
        "Confidence Code": 10,
        "BEMFAB": "M",
        "DUNS #": "10-223-5004"
    },
    "Appended Data": {
        "Company Name": "Westrock Mwv, LLC",
        "Secondary Business Name": "",
        "Address Source": "",
        "Street Address": "501 S 5TH St",
        "City": "Richmond",
        "State/Province": "VA",
        "ZIP Code": "23219-0501",
        "Country": "USA",
        "Phone": "8044441000",
        "URL": "www.westrock.com",
        "CEO Title": "President",
        "CEO First Name": "John",
        "CEO Last Name": "Luke",
        "CEO Name": "John A Luke Jr",
        "Line of Business": "Paper; coated and laminated packaging,nsk",
        "Location Type": "Headquarters",
        "Year Started": "2015",
        "Employees on Site": "741",
        "Employees Total": "15,000",
        "Sales Volume in US$": "101,818,632",
        "4 Digit SIC 1": "2671",
        "4 Digit SIC 1 Description": "Paper; Coated and Laminated Packaging",
        "4 Digit SIC 2": "2678",
        "4 Digit SIC 2 Description": "Stationery Products",
        "8 Digit SIC 1": "26710000",
        "8 Digit SIC 1 Description": "Paper; coated and laminated packaging",
        "8 Digit SIC 2": "26780000",
        "8 Digit SIC 2 Description": "Stationery products",
        "NAICS 1 Code": "322220",
        "NAICS 1 Description": "Paper Bag and Coated and Treated Paper Manufacturing",
        "NAICS 2 Code": "322230",
        "NAICS 2 Description": "Stationery Product Manufacturing",
        "Subsidiary Indicator": "subsidiary site",
        "Global Ult Indicator": "N",
        "Global Ult DUNS #": "08-109-2578",
        "Global Ult Bus. Name": "Westrock Company",
        "Global Ult State/Province": "GA",
        "Global Ult Country": "USA",
        "Domestic Ult DUNS #": "08-109-2578",
        "Domestic Ult Name": "Westrock Company",
        "Domestic Ult State/Province": "GA",
        "Domestic Ult Country": "USA",
        "Parent Ult DUNS #": "07-987-8374",
        "HQ Ult DUNS #": "",
        "HQ/Parent Ult Bus. Name": "Wrkco Inc.",
        "HQ/Parent State/Province": "GA",
        "HQ/Parent Country": "USA",
        "Hierarchy Code": "03",
        "# of Family Members": "1046"
    }
}
```

1.3 Test Endpoint (Sandbox) for the BizAPI Service

A Test Endpoint has been created to allow Users to submit fake responses for testing with Third Party Applications without having to consume existing credits.

POST https://www.naics.com/wp-json/naicsapi/v2/cosearchtest

The Sandbox requires basic authentication just like the live endpoint.

To Yield a Successful append result, the POST body can reflect any of the Input field combinations below (no need to fill the Fields in `Red` with Real Company Data) to yield a successful match.

DUNS Match

```
{
    "duns": "10-223-5004"
}
```

Standard Match

```
{
    "companyName": "Westrock Mwv, LLC",
    "address": "501 S 5TH St",
    "city": "Richmond",
    "state": "VA",
    "postalCode": "23219",
    "country": "US",
    "phone": "8044441000"
}
```

Loose Match (you can opt to include or exclude empty fields in your request)

```
{
    "companyName": "Westrock Mwv, LLC",
    "address": "",
    "city": "",
    "state": "VA",
    "postalCode": "",
    "country": "US",
    "phone": "8044441000"
}
```

Name Match (No Country Needed if US Location)

```
{
    "companyName": "Westrock Mwv, LLC",
}
```

URL Match

```
{
    "url": "www.westrock.com"
}
```

Phone Match (will not trigger if Company Name, URL, or DUNS is present in request)

```
{
    "phone": "8044441000"
}
```

To Yield an Unsuccessful append result, the POST body **must contain the word “bad” in any of the input fields**. The Unsuccessful Response will be in the same format as in the Live Endpoint. If “bad” appears inside of a word (for example Carlsbad Market), this will also trigger an Unsuccessful append result.

DUNS Match

```
{
    "duns": "bad"
}
```

Standard Match

```
{
    "companyName": "bad",
    "address": "501 S 5TH St",
    "city": "Richmond",
    "state": "VA",
    "postalCode": "23219",
    "country": "US",
    "phone": "8044441000",
}
```

Loose Match (you can opt to include or exclude empty fields in your request)

```
{
    "companyName": "bad",
    "address": "",
    "city": "",
    "state": "VA",
    "postalCode": "23219",
    "country": "US",
    "phone": "9999999999"
}
```

Name Match (No Country Needed if US Location)

```
{
    "companyName": "bad"
}
```

URL Match

```
{
    "URL": "bad"
}
```

Phone Match (will not trigger if Company Name, URL, or DUNS is present in request)

```
{
    "phone": "bad"
}
```

1.4 Record Layout Types

Each Record Layout utilizes a unique price tier. Contact APICloudSolutions@NAICS.com or your NAICS API Account Contact to acquire Pricing.

Each Successful 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
Street Address	CEO First Name
City	CEO Last Name
State/Province	CEO Name (full)
ZIP Code	Line of Business
Country	Location Type
	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 in US$

Prospecting Layout

Includes *NAICS & SIC* + *Telemarketing* content PLUS:

Employees on Site
Employees Total

Sales Volume in US$
URL (when available)

Prospecting with Linkage Layout

Includes *NAICS & SIC* + *Telemarketing* + *Prospecting* content PLUS:

Fields Returned	**Explanation of Fields**
Subsidiary Indicator	Indicates if Appended Record is/is not 51%+ own by a Parent.
Global Ult Indicator	Indicates if Appended Record is/is not the Global Ult Record.
Global Ult DUNS # Global Ult Bus. Name Global Ult State/Province Global Ult Country	The Global Ultimate Company is the highest family member in the Corporate Family tree, independent of Country Location. A subject may be its own Global Ultimate. To gain Company Size Details of the Global Ult, submit a request based on the Global Ult DUNS #
Domestic Ult DUNS # Domestic Ult Bus. Name Domestic Ult State/Province Domestic Ult Country	The Domestic Ultimate Company is the highest family member in the same country as the Appended business record. A subject may be its own Domestic Ultimate. To gain Company Size Details of the Domestic Ult, submit a request based on the Domestic Ult DUNS #
HQ Ult DUNS # OR Parent Ult DUNS # HQ/Parent Ult Bus. Name HQ/Parent State/Province HQ/Parent Country	If Appended Record is a Headquarters or Single Location, then these fields reflect the Parent Company which has more than 50%+ ownership of the Appended business record. If the Appended Record is a Branch Location, then these fields reflect the Headquarter Location the Branch reports to.
Hierarchy Code	The hierarchy code is a two-digit field which determines the records relative position in a family tree by indicating its relationship to other records. The hierarchy code functions in the following way- Global Ultimates have a hierarchy code of- 01. Subsidiaries have a hierarchy code of one greater than their parents. Branches have a hierarchy code equal to their headquarters.
# Of Family Members	The total number of family members within a family tree. This count includes the global ultimate itself, all global ultimates' branch locations, all subsidiaries, and all subsidiary branch locations.

Review the Data Dictionary for a more complete explanation of all available fields.
https://www.naics.com/bizapi-documents-v2/

1.5 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 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 Company Name and Address fields, which in fact is not the customer's address.

Some common examples of this issue:

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 correlates to a Different Business then provided in the record. This frequently occurs where the User sells through dealers and/or Value-Added Resellers (VARs). It can also occur if the Business on Record provides the Address of the Third Party they are representing instead of their own address.

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:

- No layout needs to be specified when submitting a request. Layout is established upon API Credential Activation.
- Each Field can handle over 200 characters. 65 characters or less is generally better for matching.
- 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.
- "companyName": "<company name>", should contain only One company name. If working with more than One name for the same organization, consider running a second round using the Additional Name if the first submission does not append.
- "*address*": "<*address*>" Should only Contain the Street Address OR a PO Box #.
- "*phone*": "<*phone*>" can handle most Standard Phone Formats. Do not include Extensions. Use a string of numbers with no Special characters to ensure the highest chance for a match.
- If a submission has a Zip code but lacks City and/or State information, our system will use the Primary Postal City and State linked to the Zip code to allow for an append to be possible.
- 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

To get the most precise match, DUNS Number is the best matching field. Submitting Company Name, full address, and phone will yield the second most precise match. The other Match methods rely on looser information, so they prioritize the Largest Location (based on sales/employee figures) that meet the input criteria provided. Finding the largest location might be preferable to some Users even if they have Input data for a Specific Location.

If you have a significant number of Business Records that you wish to submit in quick succession, reach out to your NAICS Representative or APICloudSolutions@naics.com to discuss data challenges. We intend to offer Batch upload solutions via API down the road, however, we can assist with any large projects that arise in the interim.

2.0 Understanding Matching Data

Matching Data is included to help you understand the Result of your Matched Request. There are 7 Fields included:

```
        "Request ID": 210
        "Layout": "PL",
        "Matches Remaining": 75774,
        "Match Method": "Standard",
        "Match Grade": "AAAAAZA",
        "Confidence Code": 10,
        "BEMFAB": "M",
        "DUNS #": "10-223-5004",
```

2.1 Request ID

The Request ID is a Unique Number Sequence that is associated with each request you make. Should requests need to be reviewed, the Request ID will help us to compare our Tracking data with yours. We encourage you to retain this information in your records in case we ever need to review tracking.

2.2 Layout

The Layout indicates which Record Layout your account is established with.

NA	NAICS & SIC	**SA**	ETM w/Sales
TA	Telemarketing	**PA**	Prosect
EA	ETM w/Emp	**PL**	Prospect w/Linkage

2.2 Matches Remaining

Matches Remaining indicates how many Unused Credits you have remaining in your account. When this number reaches 0, you will no longer be able to submit new requests. To refresh your account with additional credits, reach out to your NAICS Representative, APICloudSolutions@naics.com, or call 973-625-5626. Please reference your API account and indicate how many additional credits you would like applied to your account.

2.3 Match Method

Match Method indicates which of the 6 Match Methodologies were used in the request. You can learn more about the Match Methods in section 0.2 How does Matching work?

2.4 Match Grade

The Match Grade is a 7 Letter String that appears when a Successful Standard, Loose, and Name Match Occurs. Each letter in the string indicates how successfully a piece of the Input Data was matched to the Database Record. DUNS, URL, and Phone Match will not return a Match Grade.

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. If you are utilizing a Loose Match or Name match, you can expect many of the characters to reflect a Z.

2.5 Confidence Code

Confidence codes scale from 0 to 10 and indicate the strength of match achieved between your input data and the Record we appended.

- Successful **DUNS Matches, URL Matches, and Phone Matches** will always reflect a 10.
- Successful **Loose Matches and Name Matches** will always reflect a Confidence Level of 8.
- Successful **Standard Matches** will return a Confidence Level of 7 or Higher based on how well each Data Element is reflected in the Match Grade.

Records with a Confidence Level of 7 or Higher are strong matches. Anything lower reflects enough conflicting information between the two records to indicate the Correct Organization was not found, so we do not return them in the API.

2.6 Bemfab

The Bemfab indicator gives insight about the Marketability of the Appended Record. 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. Cannot use data to market to company but can append Industry code.

O = Unconfirmed record or confirmed as Out of business.

2.7 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.

3.0 CODE SNIPPETS

3.1 Code Snippet for PHP

```
<?php
        $username = "xxx"; //Replace xxx with your username.
        $password = "xxx"; //Replace xxx with your 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'],
                        'duns' => $_POST['duns'],
                        'url' => $_POST['url'],
                );

                // Get cURL resource
                $ch = curl_init();

                // Set url
                curl_setopt($ch, CURLOPT_URL, 'https://www.naics.com/wp-json/naicsapi/v2/cosearch');
                // Set method
                curl_setopt($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>
                        DUNS:<br>
                        <input type='text' name='DUNS'><br>
                        URL:<br>
                        <input type='text' name='URL'><br>
```

```
				<input type='submit' value='Submit'>
			</form>
			<p><?php echo $status; ?></p>
			<?php echo $htmlContent; ?>
		</body>
</html>
```

3.2 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-
hwg4gsxgFZhOsEEamdOYGBf13FyQuiTwlAQgxVSNgt4=" 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/v2/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>
                    DUNS:<br>
                    <input type='text' name='DUNS'><br>
                    URL:<br>
                    <input type='text' name='URL'><br>
                    <input id="submitBtn" type='submit' value='Submit'>
                </form>
<%
 end if
%>
                <p id="requestStatus"></p>
                <ul id="responseList"></ul>
        </body>
</html>
```

3.3 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-
hwg4gsxgFZhOsEEamdOYGBf13FyQuiTwlAQgxVSNgt4=" 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 = "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/v2/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);
```

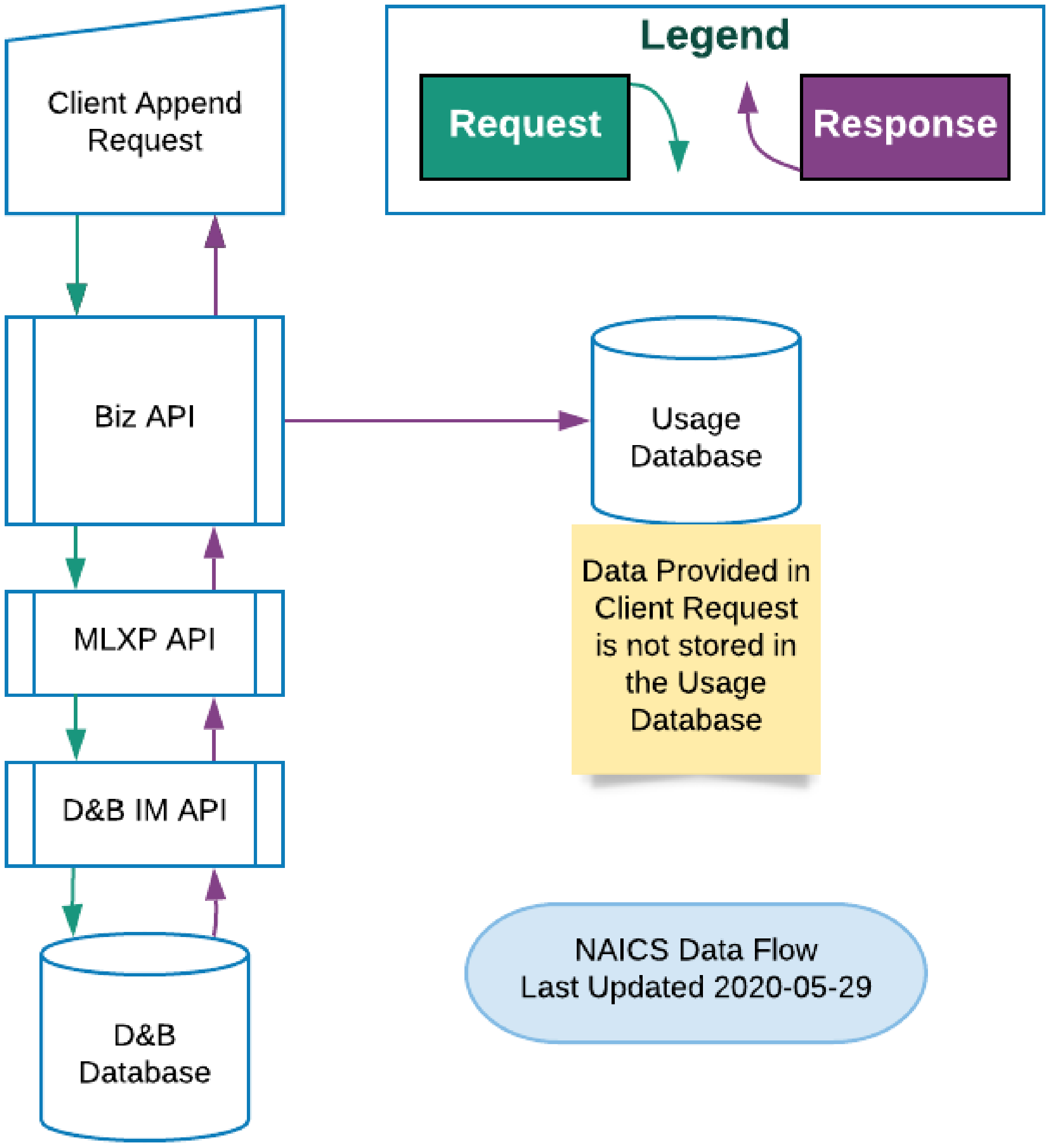
```
                        $.each(failResponse, function(key, item){
                            if (key != "data"){
                                htmlContent += "<li>";
                                htmlContent += key + ": <strong>" + item +
"</strong>";

                                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>
        <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>
            DUNS:<br>
            <input type='text' name='DUNS'><br>
            URL:<br>
            <input type='text' name='URL'><br>
            <input id="submitBtn" type='submit' value='Submit'>
        </form>
        <p id="requestStatus"></p>
        <ul id="responseList"></ul>
    </body>
</html>
```

4.0 Data Flow Architecture

The following Architecture demonstrates the flow of Data from start to finish of the Request. Input data is anonymized to ensure the security of our clients.


Legend
Request
Response
Client Append Request
Biz API
Usage Database
Data Provided in Client Request is not stored in the Usage Database
MLXP API
D&B IM API
D&B Database
NAICS Data Flow
Last Updated 2020-05-29