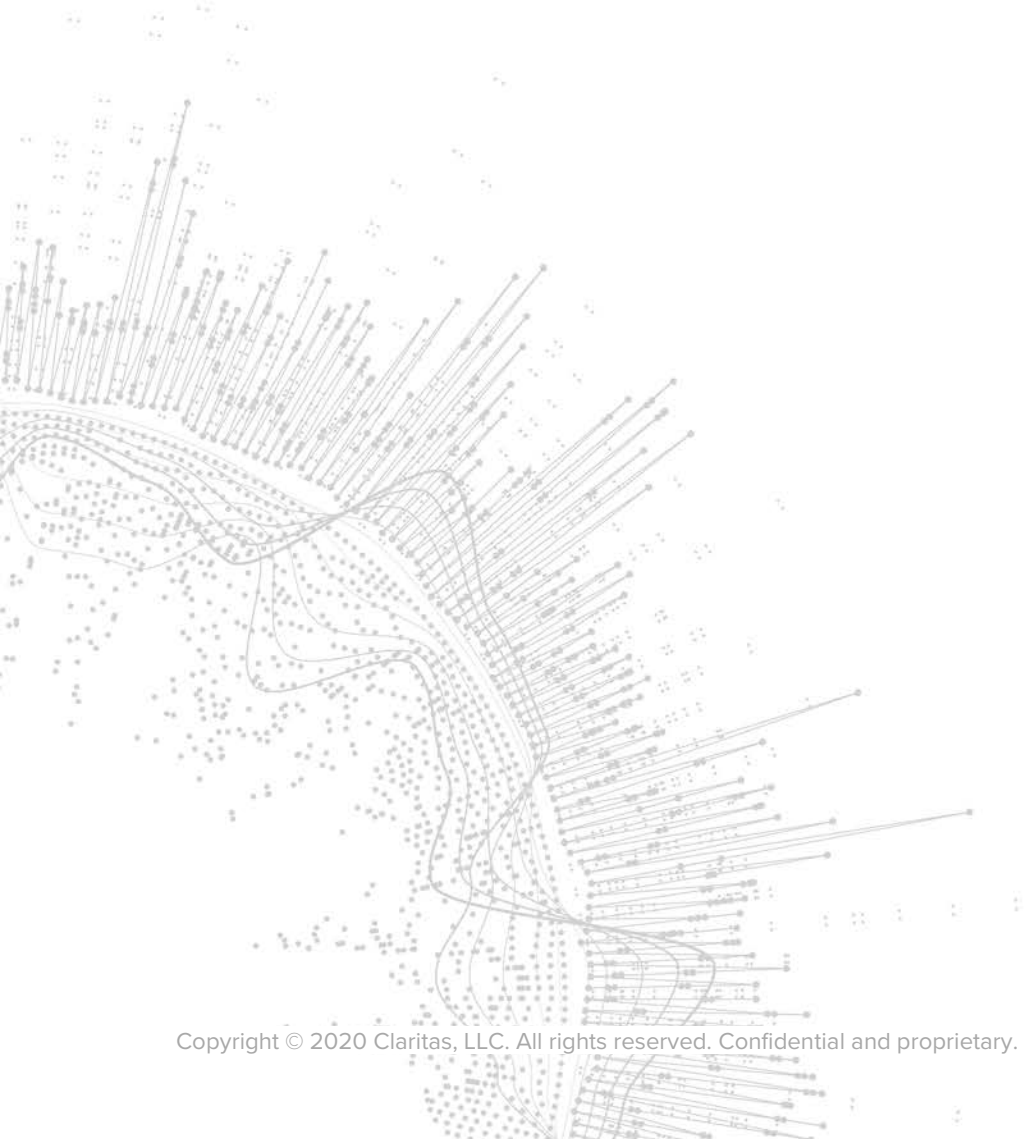
CLARITAS 360 WEB SERVICES GUIDE

May 2020



identify
deliver
optimize

Claritas 360 Web Services Guide

Contents

INTRODUCTION

Claritas 360 web services allow you to access and communicate with the Claritas 360 modules via application programming interface (API) calls without accessing via the standard login and user interface.

This document describes the Standard Operating Procedures (SOP) for the actions necessary to connect and run web services for Claritas 360. It includes the following:

- List of web services
- Parameters and their values
- Sample screens showing the Uniform Resource Locator (URL) structure, query parameters, headers, and JavaScript Object Notation (JSON) input/output

ACCESS CLARITAS 360 WEB SERVICES

1. Access Claritas 360 via this URL: https://claritas360.claritas.com.
   For testing, use https://claritas360stg.claritas.com.
2. Contact your Claritas account representative to add web services for your company and/or for users. You should receive your Claritas 360 ID and password from your account representative:
3. When the above credentials are received, you can start using the Claritas 360 API.

Important! When the access token expires, you need to procure another token via a web service call.

COMMON HEADERS AND URL PARAMETERS

These are headers and URL parameters that you can use across most Claritas 360 web service calls. Headers and URL parameters specific to a particular web service are mentioned in the section specific for that web service.

Types of Input for Web Service Calls

Input as a JSON

A JavaScript Object Notation (JSON) input with JSON output.

Header Type: *Content-Type*
Value: *application/json*

Input as a File

A multipart input that includes inputs through the URL and a CSV and TXT file import.

Header Type: *Content-Type*
Value: *multipart/form-data*

AUTHENTICATION AND AUTHORIZATION TO USE WEB SERVICES

This section details on how to procure authorization for the Claritas 360 web services system and generate an access token used in the web service calls. Sample screens are based using Postman and Java® code; yours could be different.

Procure Authorization

Request a **REFRESH_TOKEN** from your Claritas account representative and get the **refresh_token** for accessing the File Enhancement web services.

Authentication

This process creates an access token to allow the Claritas 360 system to identify the user of the web service call.

Service URL: https://claritas360.claritas.com/smsapi/authentication/auth/webservice/login

For Testing URL: https://claritas360stg.claritas.com/smsapi/authentication/auth/webservice/login

Headers: refresh_token which you received from your Claritas representative.

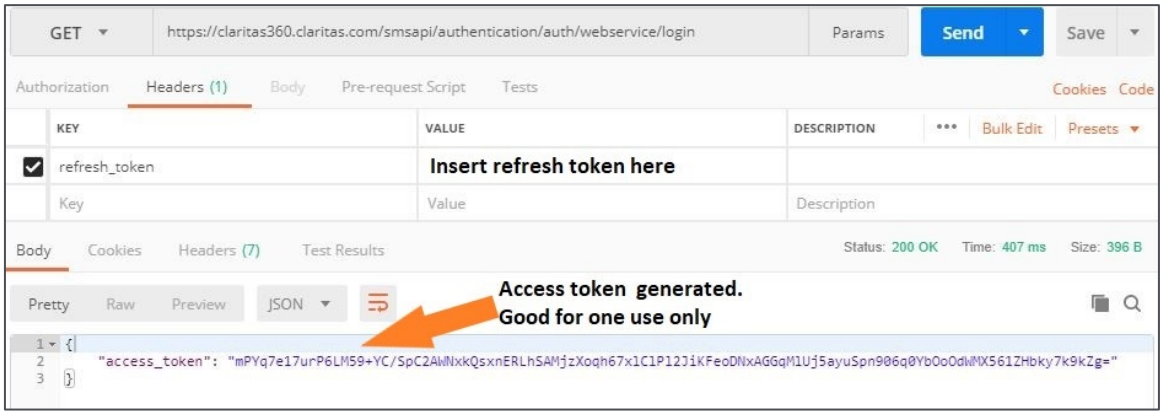
Service Method: GET

Output: access_token

Steps (via POSTMAN):

1. Select **Headers** and then do the following:
2. Add **refresh_token** under **Key,** and then copy the **access_token** output from the **Authentication call** to **Value**.

3. Click **Send**.



Access token from refresh token

4. In the **Body**, you receive an **access_token** used in other web service calls to provide identification to the Claritas 360 system. Copy the contents between the double quotes to the **access_token** value field mentioned in other calls. This is a one-time use token and needs to be regenerated after every use.

Sample Java Code Using Spring's RestTemplate

```
ResponseEntity<String> getAccessToken(String refreshToken) {
   RestTemplate restTemplate = new RestTemplate();

   HttpHeaders headers = new HttpHeaders();
   headers.setContentType(MediaType.APPLICATION_JSON);
   headers.set("refresh_token", refreshToken);
   HttpEntity<String> entity = new HttpEntity<>("parameters", headers);

   String resourceUrl = "https://claritas360.claritas.com/smsapi/authentication/auth/webservice/login";

   return restTemplate.exchange(resourceUrl, HttpMethod.GET, entity, String.class);
}
```

For Testing

```
ResponseEntity<String> getAccessToken(String refreshToken) {
   RestTemplate restTemplate = new RestTemplate();

   HttpHeaders headers = new HttpHeaders();
   headers.setContentType(MediaType.APPLICATION_JSON);
   headers.set("refresh_token", refreshToken);
   HttpEntity<String> entity = new HttpEntity<>("parameters", headers);

   String resourceUrl = "https://claritas360stg.claritas.com/smsapi/authentication/auth/webservice/login";

   return restTemplate.exchange(resourceUrl, HttpMethod.GET, entity, String.class);
}
```

Response

response =>

{ “accessToken =
“pA10gXy2flqSk3Rijd6j4cV9/0hC7o9b6FoEMSmVdaX+VIkVel51KeqwRq6w0xcO6yJAYcZuvRQbJ0ivOOw4y
KCd5P2KMYi1Rr5GOTvqoe0=” }

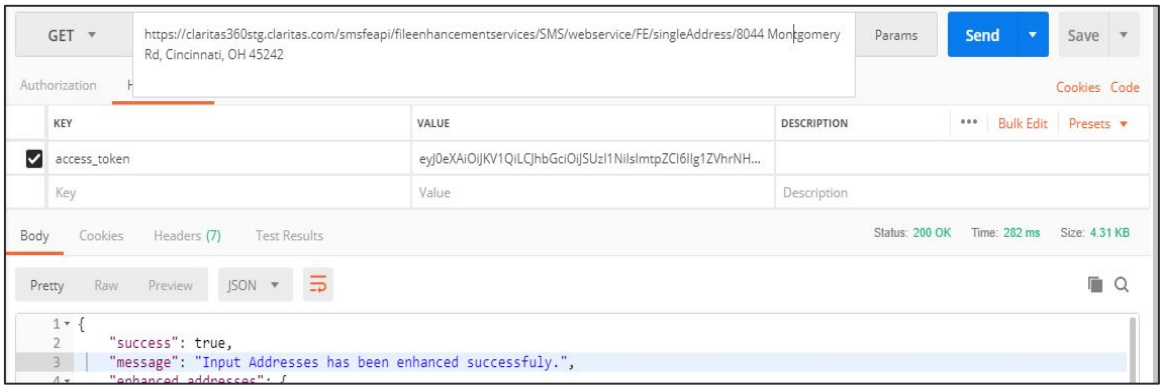
FILE ENHANCEMENT SERVICES

The File Enhancement services enhance addresses with licensed geocodes, segmentation codes, and other variables of interest. If you are not licensed for a data element, an error message will appear. You may also receive summary statistics for the geography and segmentation coding.

Single Address Look Up

This service allows enhancement of a single address. Follow the steps below to run the single address look up.

1. Generate an access token. Refer to the "Authentication and Authorization to Use Web Services – Authentication" section for instructions on page 5.
2. Run the **GET** method on the following URL:
   - https://claritas360.claritas.com/smsfeapi/fileenhancementservices/SMS/webservice/FE/singleAddress/{address}, replacing {address} with an address e.g., 123 Main St, Buffalo, NY 14217.
   - For Testing: https://claritas360stg.claritas.com/smsfeapi/fileenhancementservices/SMS/webservice/FE/singleAddress/{address}, replacing {address} with an address e.g., 123 Main St, Buffalo, NY 14217.
3. Select **Headers**, add **access_token** under **Key**, and then copy the **access_token** output from the **Authentication call** to **Value**.
4. Click **Send**.



Screen showing the GET method

This results in a JSON output of an enhanced address. Here is sample JSON output for this call: FESingleAddressJSONOutput

Sample Java Code Using Spring's RestTemplate

```
ResponseEntity<String> getSingleAddress(String accessToken) {
   RestTemplate restTemplate = new RestTemplate();
```

```
    HttpHeaders headers = new HttpHeaders();
    headers.setContentType(MediaType.APPLICATION_JSON);
    headers.set("access_token", accessToken);
    HttpEntity<String> entity = new HttpEntity<>("parameters", headers);

    String resourceUrl =
"https://claritas360.claritas.com/smsfeapi/fileenhancementservices/SMS/webservice/FE/singleAddress/8044
Montgomery Rd, Cincinnati, OH 45242";
    return restTemplate.exchange(resourceUrl, HttpMethod.GET, entity, String.class);
}
```

For Testing

```
ResponseEntity<String> getSingleAddress(String accessToken) {
    RestTemplate restTemplate = new RestTemplate();

    HttpHeaders headers = new HttpHeaders();
    headers.setContentType(MediaType.APPLICATION_JSON);
    headers.set("access_token", accessToken);
    HttpEntity<String> entity = new HttpEntity<>("parameters", headers);

    String resourceUrl =
"https://claritas360stg.claritas.com/smsfeapi/fileenhancementservices/SMS/webservice/FE/singleAddress/80
44 Montgomery Rd, Cincinnati, OH 45242";
    return restTemplate.exchange(resourceUrl, HttpMethod.GET, entity, String.class);
}
```

Response


```
response => {"success": true,"message": "Input Addresses has been enhanced successfuly.",
            "enhanced_addresses": {
                        "success": true,
                        "message": "Loaded Data",
                        "total": 1,
                        "data": [[{
                                    "inputAddress": "8044 Montgomery Rd",
                                    "inputState": null,
                                    "inputCity": null,
                                    "inputZIP": null,
                                    "inputLastLine": " Cincinnati, OH 45242",
                                    "inputBlockgroup": null,
                                    "standardizedAddress": "8044 MONTGOMERY RD",
                                    "standardizedState": "OH",
                                    "standardizedZip": "45236",
                                    "latitude": "39.202164",
                                    "longitude": "-84.371734",
                                    "blockGroup": "390610240022",
                                    "matchCode": "S90",
                                    "matchCodeDesc": "Match found in USPS data.ZIP and ZIP + 4 changed.No change in address
line.",
```

```
                "zip": "45236",
                "zipName": null,
                "zip4": "452362919",
                "zip6": "45236291999",
                "locationCode": "AS0",
                "locationCodeDesc": "House range address geocode. This is the most accurate street interpolated geocode available. Best location.",
                "carrierRouteCode": "C072",
                "standardizedCity": "CINCINNATI",
                "matchFlag": "Exact",
                "segAppend": [{

                },
                {
                        "PRIZM Premier SEGMENT NAME": "Beltway Boomers",
                        "PRIZM Premier SEGMENT DESC": "The members of the postwar Baby Boom are all grown up. One segment of this huge cohort, college-educated, upper-middle-class, and home-owning, is found in Beltway Boomers. Like many of their peers who married late, many of these Boomers are still raising children in comfortable suburban subdivisions while beginning to plan for their own retirement.",
                        "PRIZM Premier SEGMENT FLAG (ZP4)": "3",
                        "PRIZM Premier SEGMENT (ZP4)": "16"
                },
                {
                        "PRIZM Premier SEGMENT FLAG (BGR)": "B",
                        "PRIZM Premier SEGMENT NAME": "Beltway Boomers",
                        "PRIZM Premier SEGMENT DESC": "The members of the postwar Baby Boom are all grown up. One segment of this huge cohort, college-educated, upper-middle-class, and home-owning, is found in Beltway Boomers. Like many of their peers who married late, many of these Boomers are still raising children in comfortable suburban subdivisions while beginning to plan for their own retirement.",
                        "PRIZM Premier SEGMENT (BGR)": "16"
                },
                {
                        "PRIZM Premier SEGMENT FLAG (ZIP)": "Z",
                        "PRIZM Premier SEGMENT NAME": "Kids & Cul-de-Sacs",
                        "PRIZM Premier SEGMENT DESC": "Upper-middle-class, suburban, married couples with children - that's the skinny on Kids & Cul-de-Sacs, an enviable lifestyle of large families in recently built subdivisions. This segment is a refuge for college-educated, white-collar professionals with administrative jobs and upper-middle-class incomes. Their nexus of education, affluence, and children translates into large outlays for child-centered products and services.",
                        "PRIZM Premier SEGMENT (ZIP)": "14"
                },
                {

                },
                {

                }],
                "scoreAppend": [{
                        "Claritas Net Worth Indicators SCORE FLAG (ZP6)": "3",
                        "Claritas Net Worth Indicators SCORE (ZP6)": "06"
                },
                {
```

```
                        "Claritas Income Producing Assets Indicators SCORE FLAG (ZP6)": "3",
                        "Claritas Income Producing Assets Indicators SCORE (ZP6)": "05"
                    },
                    {
                        "Claritas Consumer Score - Heavy Technology Usage SCORE FLAG (ZP6)": "B",
                        "Claritas Consumer Score - Heavy Technology Usage SCORE (ZP6)": "44"
                    },
                    {
                        "Claritas Consumer Score - Technology Adoption SCORE (ZP6)": "44",
                        "Claritas Consumer Score - Technology Adoption SCORE FLAG (ZP6)": "B"
                    }],
                    "geoAppend": {
                        "ZIP": "45243",
                        "Census Place Name": "Kenwood CDP",
                        "State Name": "Ohio",
                        "Minor Civil Division (MCD)": "3906175973",
                        "County": "39061",
                        "County Name": "Hamilton County",
                        "Designated Market Area (DMA) Name": "Cincinnati, OH",
                        "State Code": "39",
                        "Combined Statistical Areas (CSA)": "178",
                        "Census Tract": "39061024002",
                        "Core Based Statistical Area (CBSA)": "17140",
                        "Minor Civil Division (MCD) Name": "Sycamore township",
                        "Census Block Group": "390610240022",
                        "Nielsen Designated Market Area (DMA)": "515",
                        "Census Place": "3939914",
                        "Three Digit ZIP (TDZ)": "452",
                        "Core Based Statistical Area (CBSA) Name": "Cincinnati, OH-KY-IN",
                        "Combined Statistical Area (CSA) Name": "Cincinnati et al, OH-KY-IN"
                    },
                    "tdzcode": "452"
                }]]
        },
        "rejected_addresses": {
            "success": true,
            "message": "No Records Found",
            "total": 0,
            "data": []
        }
}
```


Modify URL to Change Address

To change the location address, simply replace the address with the new one as shown below.

The first example shows a URL containing a single address. The second one shows the changed address (i.e., the bolded text).

https://claritas360.claritas.com/smsfeapi/fileenhancementservices/SMS/webservice/FE/singleAddress/8044 Montgomery Rd, Cincinnati, OH 45242

https://claritas360.claritas.com/smsfeapi/fileenhancementservices/SMS/webservice/FE/singleAddress/456**Scranton Rd, San Diego, CA 92121**

For Testing

https://claritas360stg.claritas.com/smsfeapi/fileenhancementservices/SMS/webservice/FE/singleAddress/8044 Montgomery Rd, Cincinnati, OH 45242

https://claritas360stg.claritas.com/smsfeapi/fileenhancementservices/SMS/webservice/FE/singleAddress/456**Scranton Rd, San Diego, CA 92121**

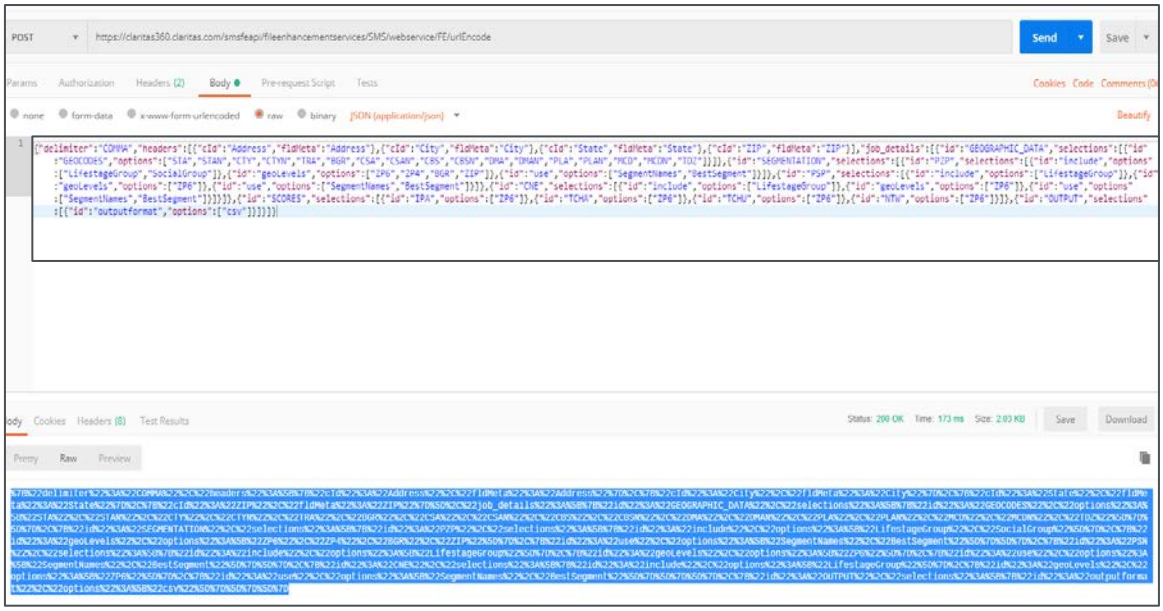
File Enhancement Batch

This service allows enhancement of several addresses via a CSV and TXT file upload. Follow the steps below to run File Enhancement Batch.

File Enhancement Batch Service with CSV and TXT File

1. The uploaded file should contain the following restrictions:
   - The title column should not have any spaces.
   - The title column should match the userOptions.
2. To generate the proper userOptions in the JSON format. Please refer to the Prepare User Options for File Enhancement Batch section.

3. Encode userOption. This steps only apply for postman since Java auto encode url.
    - https://claritas360.claritas.com/smsfeapi/fileenhancementservices/SMS/webservice/FE/urlEncode
    - method: post.
    - Header: access_token: your access token
    - Header: Content-Type: application/json
    - Body: your request json

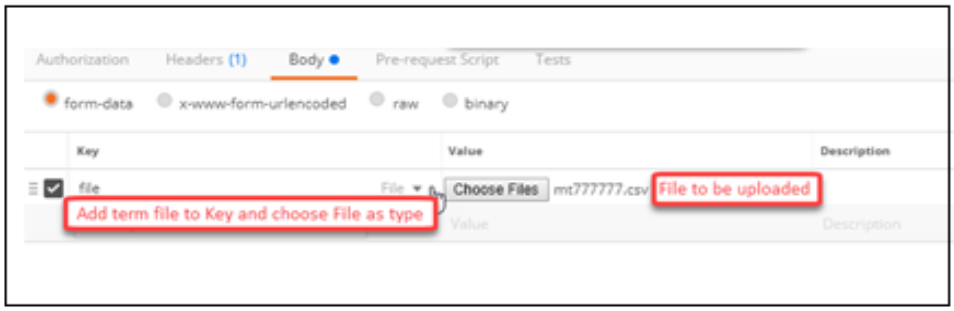


4. Run the **POST** method on the following URL:
    - https://claritas360.claritas.com/smsfeapi/fileenhancementservices/SMS/webservice/FE/fileUpload? userOptions=<Encoded **User options**>
    - For testing: https://claritas360.claritas.com/smsfeapi/fileenhancementservices/SMS/webservice/FE/fileUpload? userOptions=<Encoded **User options**>

Sample URL with user options

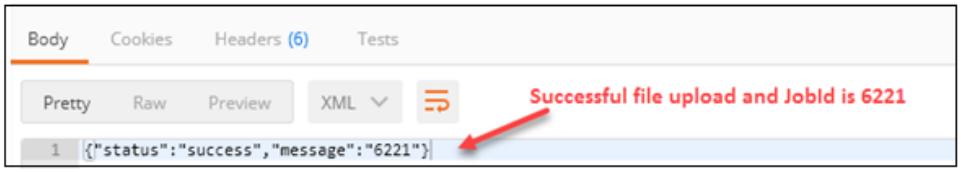
5. Select **Headers**, add **access_token** under **Key**, and then copy the **access_token** output from the **Authentication call** to **Value.**

6. Select **Body** and then do the following:
7. Under **form-data** type, add **file** under **Key**.
8. Change the type of input from **TEXT** to **File** to allow a file upload.
9. In **Value**, browse and select your address file.



Screen showing how to add term file and selecting file to upload

10. Click **Send.** If successful, a message and a Job ID appear as shown below.



Screen showing the “success” message and the Job ID

CSV Content

Address,City,State,ZIP

319 Hickory Trl,Crystal Spg,PA,15536

8349 S Valley Rd,Crystal Spg,PA,15536

2439 S Valley Rd,Crystal Spg,PA,15536

935 Barton Rd,Crystal Spg,PA,15536

Sample Java Code Using Spring's RestTemplate

```
ResponseEntity<String> postBatchFileEnhancement(String accessToken) {
   RestTemplate restTemplate = new RestTemplate();

   MultiValueMap<String, Object> bodyMap = new LinkedMultiValueMap<>();
   bodyMap.add("file", new ClassPathResource("29k_dc_dma_addresses.csv"));

   HttpHeaders headers = new HttpHeaders();
   headers.setContentType(MediaType.MULTIPART_FORM_DATA);
   headers.set("access_token", accessToken);

   HttpEntity<MultiValueMap<String, Object>> requestEntity = new HttpEntity<>(bodyMap, headers);

   URI uri = UriComponentsBuilder.newInstance().scheme("https").host("claritas360.claritas.com")
         .path("/smsfeapi/fileenhancementservices/SMS/webservice/FE/fileUpload")
```

```
        .queryParam("userOptions",
"{\"delimiter\":\"COMMA\",\"headers\":[{\"cId\":\"Address\",\"fldMeta\":\"Address\"},{\"cId\":\"City\",\"fldMeta\":\"City\"},{\"cId\":\"State\",\"fldMeta\":\"State\"},{\"cId\":\"ZIP\",\"fldMeta\":\"ZIP\"}],\"job_details\":[{\"id\":\"GEOGRAPHIC_DATA\",\"selections\":[{\"id\":\"GEOCODES\",\"options\":[\"STA\",\"STAN\",\"CTY\",\"CTYN\",\"TRA\",\"BGR\",\"CSA\",\"CSAN\",\"CBS\",\"CBSN\",\"DMA\",\"DMAN\",\"PLA\",\"PLAN\",\"MCD\",\"MCDN\",\"TDZ\"]}]},{\"id\":\"SEGMENTATION\",\"selections\":[{\"id\":\"PZP\",\"selections\":[{\"id\":\"include\",\"options\":[\"LifestageGroup\",\"SocialGroup\"]},{\"id\":\"geoLevels\",\"options\":[\"ZP6\",\"ZP4\",\"BGR\",\"ZIP\"]},{\"id\":\"use\",\"options\":[\"SegmentNames\",\"BestSegment\"]}]},{\"id\":\"PSP\",\"selections\":[{\"id\":\"include\",\"options\":[\"LifestageGroup\"]},{\"id\":\"geoLevels\",\"options\":[\"ZP6\"]},{\"id\":\"use\",\"options\":[\"SegmentNames\",\"BestSegment\"]}]},{\"id\":\"CNE\",\"selections\":[{\"id\":\"include\",\"options\":[\"LifestageGroup\"]},{\"id\":\"geoLevels\",\"options\":[\"ZP6\"]},{\"id\":\"use\",\"options\":[\"SegmentNames\",\"BestSegment\"]}]}],{\"id\":\"SCORES\",\"selections\":[{\"id\":\"IPA\",\"options\":[\"ZP6\"]},{\"id\":\"TCHA\",\"options\":[\"ZP6\"]},{\"id\":\"TCHU\",\"options\":[\"ZP6\"]},{\"id\":\"NTW\",\"options\":[\"ZP6\"]}]},{\"id\":\"OUTPUT\",\"selections\":[{\"id\":\"outputformat\",\"options\":[\"csv\"]}]}]}").build().toUri();

    return restTemplate.exchange(uri, HttpMethod.POST, requestEntity, String.class);
}
```

For Testing

```
ResponseEntity<String> postBatchFileEnhancement(String accessToken) {
    RestTemplate restTemplate = new RestTemplate();

    MultiValueMap<String, Object> bodyMap = new LinkedMultiValueMap<>();
    bodyMap.add("file", new ClassPathResource("29k_dc_dma_addresses.csv"));

    HttpHeaders headers = new HttpHeaders();
    headers.setContentType(MediaType.MULTIPART_FORM_DATA);
    headers.set("access_token", accessToken);

    HttpEntity<MultiValueMap<String, Object>> requestEntity = new HttpEntity<>(bodyMap, headers);

    URI uri = UriComponentsBuilder.newInstance().scheme("https").host("claritas360stg.claritas.com")
            .path("/smsfeapi/fileenhancementservices/SMS/webservice/FE/fileUpload")
            .queryParam("userOptions",
"{\"delimiter\":\"COMMA\",\"headers\":[{\"cId\":\"Address\",\"fldMeta\":\"Address\"},{\"cId\":\"City\",\"fldMeta\":\"City\"},{\"cId\":\"State\",\"fldMeta\":\"State\"},{\"cId\":\"ZIP\",\"fldMeta\":\"ZIP\"}],\"job_details\":[{\"id\":\"GEOGRAPHIC_DATA\",\"selections\":[{\"id\":\"GEOCODES\",\"options\":[\"STA\",\"STAN\",\"CTY\",\"CTYN\",\"TRA\",\"BGR\",\"CSA\",\"CSAN\",\"CBS\",\"CBSN\",\"DMA\",\"DMAN\",\"PLA\",\"PLAN\",\"MCD\",\"MCDN\",\"TDZ\"]}]},{\"id\":\"SEGMENTATION\",\"selections\":[{\"id\":\"PZP\",\"selections\":[{\"id\":\"include\",\"options\":[\"LifestageGroup\",\"SocialGroup\"]},{\"id\":\"geoLevels\",\"options\":[\"ZP6\",\"ZP4\",\"BGR\",\"ZIP\"]},{\"id\":\"use\",\"options\":[\"SegmentNames\",\"BestSegment\"]}]},{\"id\":\"PSP\",\"selections\":[{\"id\":\"include\",\"options\":[\"LifestageGroup\"]},{\"id\":\"geoLevels\",\"options\":[\"ZP6\"]},{\"id\":\"use\",\"options\":[\"SegmentNames\",\"BestSegment\"]}]},{\"id\":\"CNE\",\"selections\":[{\"id\":\"include\",\"options\":[\"LifestageGroup\"]},{\"id\":\"geoLevels\",\"options\":[\"ZP6\"]},{\"id\":\"use\",\"options\":[\"SegmentNames\",\"BestSegment\"]}]}],{\"id\":\"SCORES\",\"selections\":[{\"id\":\"IPA\",\"options\":[\"ZP6\"]},{\"id\":\"TCHA\",\"options\":[\"ZP6\"]},{\"id\":\"TCHU\",\"options\":[\"ZP6\"]},{\"id\":\"NTW\",\"options\":[\"ZP6\"]}]},{\"id\":\"OUTPUT\",\"selections\":[{\"id\":\"outputformat\",\"options\":[\"csv\"]}]}]}").build().toUri();

    return restTemplate.exchange(uri, HttpMethod.POST, requestEntity, String.class);
}
```

Response

```
response => {"status":"success","message":"40556"}
```

File Enhancement Job Status

1. Run the **GET** method on the following URL:
   - https://claritas360.claritas.com/smsfeapi/fileenhancementservices/SMS/webservice/FE/getJobStatus?job_id=<job_id>&downloadResults=false , replacing <job_id> with the valid ID of a job (e.g., 6221).
   - For Testing: https://claritas360stg.claritas.com/smsfeapi/fileenhancementservices/SMS/webservice/FE/getJobStatus?job_id=<job_id>&downloadResults=false , replacing <job_id> with the valid ID of a job (e.g., 6221).
2. Select **Headers**, add **access_token** under **Key**, and then copy the **access_token** output from the **Authentication call** to **Value**.
3. Click **Send**.
   If File Enhancement is still running, an "In Progress" message appears.

   If the job is completed, a completion message with the statistical status of the job appears.

   If the job errors, an "Error" message appears.

Sample Java Code Using Spring's RestTemplate

```
ResponseEntity<String> getJobStatus(String accessToken) {
        RestTemplate restTemplate = new RestTemplate();

        HttpHeaders headers = new HttpHeaders();
        headers.setContentType(MediaType.APPLICATION_JSON);
        headers.set("access_token", accessToken);
        HttpEntity<String> entity = new HttpEntity<>("parameters", headers);

        String resourceUrl =
    "https://claritas360.claritas.com/smsfeapi/fileenhancementservices/SMS/webservice/FE/getJobStatus";

        UriComponentsBuilder builder = UriComponentsBuilder.fromUriString(resourceUrl)
                .queryParam("downloadResults", false)
                .queryParam("job_id", 40556);

        return restTemplate.exchange(builder.toUriString(), HttpMethod.GET, entity, String.class);
}
```

For Testing

```
ResponseEntity<String> getJobStatus(String accessToken) {
        RestTemplate restTemplate = new RestTemplate();

        HttpHeaders headers = new HttpHeaders();
        headers.setContentType(MediaType.APPLICATION_JSON);
        headers.set("access_token", accessToken);
```

```
        HttpEntity<String> entity = new HttpEntity<>("parameters", headers);

        String resourceUrl =
    "https://claritas360stg.claritas.com/smsfeapi/fileenhancementservices/SMS/webservice/FE/getJobStatus"
    ;

        UriComponentsBuilder builder = UriComponentsBuilder.fromUriString(resourceUrl)
                .queryParam("downloadResults", false)
                .queryParam("job_id", 40556);

        return restTemplate.exchange(builder.toUriString(), HttpMethod.GET, entity, String.class);
}
```

Response (Sample with ALL options shown)

response => {"job_id":112544,"message":"Job has been completed successfully.","total_input_records":29651,"success":true,"job_status":"COMPLETED","match_codes":[{"code":"S80","desc":"Match found in USPS data.ZIP + 4 changed.No change in address line.","count":29234},{"code":"SA0","desc":"Match found in USPS data.City and ZIP + 4 changed.No change in address line.","count":380},{"code":"A88","desc":"Match to an alias name record.ZIP + 4 changed.Street name changed.","count":9},{"code":"A82","desc":"Match to an alias name record.ZIP + 4 changed.Predirectional changed.","count":7},{"code":"S88","desc":"Match found in USPS data.ZIP + 4 changed.Street name changed.","count":4},{"code":"A81","desc":"Match to an alias name record.ZIP + 4 changed.Street type changed.","count":3},{"code":"S81","desc":"Match found in USPS data.ZIP + 4 changed.Street type changed.","count":3},{"code":"E020","desc":"No matching streets found in directory.","count":3},{"code":"E022","desc":"No matching segments.","count":3},{"code":"T80","desc":"Match to the street network file.ZIP + 4 changed.No change in address line.","count":1},{"code":"A89","desc":"Match to an alias name record.ZIP + 4 changed.Street name and street type changed.","count":1},{"code":"E027","desc":"Invalid directional attempted.","count":1},{"code":"S86","desc":"Match found in USPS data.ZIP + 4 changed.Predirectional and postdirectional changed.","count":1},{"code":"A80","desc":"Match to an alias name record.ZIP + 4 changed.No change in address line.","count":1}],"match_accuracy_rates":{"Reject":{"count":7,"percentage":0.02},"All Possibilities":{"count":0,"percentage":0},"Best Match":{"count":384,"percentage":1.3},"Exact Match":{"count":29260,"percentage":98.68}},"segment_match_rates":[{"seg_system_id":"PZP","seg_system_name":"PRIZM Premier","seg_geolevel_match_rates":{"ZIP":{"Assigned at ZIP+4 Level":{"count":0,"percentage":0},"Assigned as Non-Residential PO Box at the ZIP level":{"count":0,"percentage":0},"Assigned as Non-Residential at the Block Group Level":{"count":0,"percentage":0},"Assigned as Residential PO Box at ZIP+4 level":{"count":0,"percentage":0},"Assigned as Residential PO Box at ZIP+6 level":{"count":0,"percentage":0},"Assigned at ZIP+6 Level":{"count":0,"percentage":0},"Assigned as Non-Residential at the ZIP level":{"count":0,"percentage":0},"Assigned at Block Group Level":{"count":0,"percentage":0},"Unmatched Records":{"count":0,"percentage":0},"Assigned as Non-Residential unable to assign at any level":{"count":0,"percentage":0},"Assigned at ZIP Level":{"count":29651,"percentage":100},"Assigned as Residential PO Box at Block Group level":{"count":0,"percentage":0},"Assigned as Residential PO Box at ZIP level":{"count":0,"percentage":0}},"ZP6":{"Assigned at ZIP+4 Level":{"count":1378,"percentage":4.65},"Assigned as Non-Residential PO Box at the ZIP level":{"count":696,"percentage":2.35},"Assigned as Non-Residential at

the Block Group Level":{"count":991,"percentage":3.34},"Assigned as Residential PO Box at ZIP+4 level":{"count":0,"percentage":0},"Assigned as Residential PO Box at ZIP+6 level":{"count":43,"percentage":0.15},"Assigned at ZIP+6 Level":{"count":26214,"percentage":88.41},"Assigned as Non-Residential at the ZIP level":{"count":49,"percentage":0.17},"Assigned at Block Group Level":{"count":196,"percentage":0.66},"Unmatched Records":{"count":0,"percentage":0},"Assigned as Non-Residential unable to assign at any level":{"count":0,"percentage":0},"Assigned at ZIP Level":{"count":76,"percentage":0.26},"Assigned as Residential PO Box at Block Group level":{"count":0,"percentage":0},"Assigned as Residential PO Box at ZIP level":{"count":8,"percentage":0.03}},"BGR":{"Assigned at ZIP+4 Level":{"count":0,"percentage":0},"Assigned as Non-Residential PO Box at the ZIP level":{"count":0,"percentage":0},"Assigned as Non-Residential at the Block Group Level":{"count":0,"percentage":0},"Assigned as Residential PO Box at ZIP+4 level":{"count":0,"percentage":0},"Assigned as Residential PO Box at ZIP+6 level":{"count":0,"percentage":0},"Assigned at ZIP+6 Level":{"count":0,"percentage":0},"Assigned as Non-Residential at the ZIP level":{"count":0,"percentage":0},"Assigned at Block Group Level":{"count":28801,"percentage":97.13},"Unmatched Records":{"count":0,"percentage":0},"Assigned as Non-Residential unable to assign at any level":{"count":0,"percentage":0},"Assigned at ZIP Level":{"count":850,"percentage":2.87},"Assigned as Residential PO Box at Block Group level":{"count":0,"percentage":0},"Assigned as Residential PO Box at ZIP level":{"count":0,"percentage":0}},"ZP4":{"Assigned at ZIP+4 Level":{"count":28002,"percentage":94.44},"Assigned as Non-Residential PO Box at the ZIP level":{"count":696,"percentage":2.35},"Assigned as Non-Residential at the Block Group Level":{"count":552,"percentage":1.86},"Assigned as Residential PO Box at ZIP+4 level":{"count":43,"percentage":0.15},"Assigned as Residential PO Box at ZIP+6 level":{"count":0,"percentage":0},"Assigned at ZIP+6 Level":{"count":0,"percentage":0},"Assigned as Non-Residential at the ZIP level":{"count":46,"percentage":0.16},"Assigned at Block Group Level":{"count":228,"percentage":0.77},"Unmatched Records":{"count":0,"percentage":0},"Assigned as Non-Residential unable to assign at any level":{"count":0,"percentage":0},"Assigned at ZIP Level":{"count":76,"percentage":0.26},"Assigned as Residential PO Box at Block Group level":{"count":0,"percentage":0},"Assigned as Residential PO Box at ZIP level":{"count":8,"percentage":0.03}}}},{"seg_system_id":"CNE","seg_system_name":"ConneXions","seg_geolevel_match_rates":{"ZP6":{"Assigned at ZIP+4 Level":{"count":1378,"percentage":4.65},"Assigned as Non-Residential PO Box at the ZIP level":{"count":696,"percentage":2.35},"Assigned as Non-Residential at the Block Group Level":{"count":991,"percentage":3.34},"Assigned as Residential PO Box at ZIP+4 level":{"count":0,"percentage":0},"Assigned as Residential PO Box at ZIP+6 level":{"count":43,"percentage":0.15},"Assigned at ZIP+6 Level":{"count":26214,"percentage":88.41},"Assigned as Non-Residential at the ZIP level":{"count":49,"percentage":0.17},"Assigned at Block Group Level":{"count":196,"percentage":0.66},"Unmatched Records":{"count":0,"percentage":0},"Assigned as Non-Residential unable to assign at any level":{"count":0,"percentage":0},"Assigned at ZIP Level":{"count":76,"percentage":0.26},"Assigned as Residential PO Box at Block Group level":{"count":0,"percentage":0},"Assigned as Residential PO Box at ZIP level":{"count":8,"percentage":0.03}}}},{"seg_system_id":"PSP","seg_system_name":"P$YCLE Premier","seg_geolevel_match_rates":{"ZP6":{"Assigned at ZIP+4 Level":{"count":1378,"percentage":4.65},"Assigned as Non-Residential PO Box at the ZIP level":{"count":696,"percentage":2.35},"Assigned as Non-Residential at the Block Group Level":{"count":991,"percentage":3.34},"Assigned as Residential PO Box at ZIP+4 level":{"count":0,"percentage":0},"Assigned as Residential PO Box at ZIP+6 level":{"count":43,"percentage":0.15},"Assigned at ZIP+6 Level":{"count":26214,"percentage":88.41},"Assigned

as Non-Residential at the ZIP level":{"count":49,"percentage":0.17},"Assigned at Block Group Level":{"count":196,"percentage":0.66},"Unmatched Records":{"count":0,"percentage":0},"Assigned as Non-Residential unable to assign at any level":{"count":0,"percentage":0},"Assigned at ZIP Level":{"count":76,"percentage":0.26},"Assigned as Residential PO Box at Block Group level":{"count":0,"percentage":0},"Assigned as Residential PO Box at ZIP level":{"count":8,"percentage":0.03}}}}],"score_match_rates":[{"scr_system_id":"NTW","scr_system_name":"Claritas Net Worth Indicators","scr_geolevel":"ZP6","match_rates":{"ZIP":{"count":76,"percentage":0.26},"Others":{"count":1787,"percentage":6.03},"ZP6":{"count":26214,"percentage":88.41},"BGR":{"count":196,"percentage":0.66},"ZP4":{"count":1378,"percentage":4.65}}},{"scr_system_id":"IPA","scr_system_name":"Claritas Income Producing Assets Indicators","scr_geolevel":"ZP6","match_rates":{"ZIP":{"count":76,"percentage":0.26},"Others":{"count":1787,"percentage":6.03},"ZP6":{"count":26214,"percentage":88.41},"BGR":{"count":196,"percentage":0.66},"ZP4":{"count":1378,"percentage":4.65}}},{"scr_system_id":"TCHU","scr_system_name":"Claritas Consumer Score - Heavy Technology Usage","scr_geolevel":"ZP6","match_rates":{"ZIP":{"count":44,"percentage":0.15},"ZP6":{"count":26257,"percentage":88.55},"BGR":{"count":1564,"percentage":5.27},"ZP4":{"count":1786,"percentage":6.02}}},{"scr_system_id":"TCHA","scr_system_name":"Claritas Consumer Score - Technology Adoption","scr_geolevel":"ZP6","match_rates":{"ZIP":{"count":44,"percentage":0.15},"ZP6":{"count":26257,"percentage":88.55},"BGR":{"count":1564,"percentage":5.27},"ZP4":{"count":1786,"percentage":6.02}}}]}

File Enhancement Job Status: Retrieve the Output File

1. Run the **GET** method on the following URL:
2. https://claritas360.claritas.com/smsfeapi/fileenhancementservices/SMS/webservice/FE/getJobStatus?job_id=<job_id> replacing <job_id> with the valid ID of a job (e.g., 6221).
3. For Testing: https://claritas360stg.claritas.com/smsfeapi/fileenhancementservices/SMS/webservice/FE/getJobStatus?job_id=<job_id> replacing <job_id> with the valid ID of a job (e.g., 6221).
4. Select **Headers**, add **access_token** under **Key**, and then copy the **access_token** output from the **Authentication call** to **Value**.
5. Click **Send**.
   If File Enhancement is still running, an “In Progress” message appears.
6. **If the job is completed,** a completion status appears and the system downloads the output file in the requested format.
7. **If the job errors,** an “Error” message appears.

Sample Java Code Using Spring's RestTemplate

```
ResponseEntity<String> getJobStatusAndOutput(String accessToken) {
   RestTemplate = new RestTemplate();

   HttpHeaders headers = new HttpHeaders();
   headers.setContentType(MediaType.APPLICATION_JSON);
   headers.set("access_token", accessToken);
   HttpEntity<String> entity = new HttpEntity<>("parameters", headers);
   String resourceUrl =
```

```
"https://claritas360.claritas.com/smsfeapi/fileenhancementservices/SMS/webservice/FE/getJobStatus";

    UriComponentsBuilder builder = UriComponentsBuilder.fromUriString(resourceUrl)
            .queryParam("job_id", 40556);

    return restTemplate.exchange(builder.toUriString(), HttpMethod.GET, entity, String.class);
}
```

For Testing

```
ResponseEntity<String> getJobStatusAndOutput(String accessToken) {
    RestTemplate = new RestTemplate();

    HttpHeaders headers = new HttpHeaders();
    headers.setContentType(MediaType.APPLICATION_JSON);
    headers.set("access_token", accessToken);
    HttpEntity<String> entity = new HttpEntity<>("parameters", headers);

    String resourceUrl =
"https://claritas360stg.claritas.com/smsfeapi/fileenhancementservices/SMS/webservice/FE/getJobStatus";

    UriComponentsBuilder builder = UriComponentsBuilder.fromUriString(resourceUrl)
            .queryParam("job_id", 40556);

    return restTemplate.exchange(builder.toUriString(), HttpMethod.GET, entity, String.class);
}
```

Response

response =>
id,u_address,u_city,u_state,u_zip,standardized_address,standardized_city,standardized_state,standardized_zip,match_flag,match_code,location_code,carrier_route_code,latitude,longitude,zip_gcode,bgr_gcode,zp4_gcode,zp6_gcode,sta_gcode,sta_geographic_name,cty_gcode,cty_geographic_name,tra_gcode,csa_gcode,csa_geographic_name,cbs_gcode,cbs_geographic_name,dma_gcode,dma_geographic_name,pla_gcode,pla_geographic_name,mcd_gcode,mcd_geographic_name,tdz_gcode,pzp_zp6_seg_id,pzp_zp6_flag,pzp_zp6_seg_name,pzp_zp6_lifestage_group_alias,pzp_zp6_lifestage_group_name,pzp_zp6_social_group_alias,pzp_zp6_social_group_name,pzp_zp4_seg_id,pzp_zp4_flag,pzp_zp4_seg_name,pzp_zp4_lifestage_group_alias,pzp_zp4_lifestage_group_name,pzp_zp4_social_group_alias,pzp_zp4_social_group_name,pzp_bgr_seg_id,pzp_bgr_flag,pzp_bgr_seg_name,pzp_bgr_lifestage_group_alias,pzp_bgr_lifestage_group_name,pzp_bgr_social_group_alias,pzp_bgr_social_group_name,pzp_zip_seg_id,pzp_zip_flag,pzp_zip_seg_name,pzp_zip_lifestage_group_alias,pzp_zip_lifestage_group_name,pzp_zip_social_group_alias,pzp_zip_social_group_name,cne_zp6_seg_id,cne_zp6_flag,cne_zp6_seg_name,cne_zp6_lifestage_group_alias,cne_zp6_lifestage_group_name,psp_zp6_seg_id,psp_zp6_flag,psp_zp6_seg_name,psp_zp6_lifestage_group_alias,psp_zp6_lifestage_group_name,ntw_zp6_score_id,ntw_zp6_flag,ipa_zp6_score_id,ipa_zp6_flag,tchu_zp6_score_id,tchu_zp6_flag,tcha_zp6_score_id,tcha_zp6_flag

1,319 Hickory Trl,Crystal Spg,PA,15536,319 HICKORY TRL,CRYSTAL SPG,PA,15536,Exact,S80,AS0,H001,40.01544,-78.179442,15536,420579602004,155366927,15536692719,42,Pennsylvania,42057,Fulton County,42057960200,942,Unassigned Area - Pennsylvania,99942,Unassigned Area - Pennsylvania,511,"Washington, DC (Hagerstown, MD)",4299999,Remainder of

Pennsylvania,4205709568,Brush Creek township,155,27,6,Big Sky Families,F2,Young Accumulators,T2,Country Comfort,28,4,Country Casuals,M2,Conservative Classics,T2,Country Comfort,46,B,Heartlanders,M3,Cautious Couples,T3,Middle America,52,Z,Simple Pleasures,M3,Cautious Couples,T3,Middle America,09,6,Big Spenders,F1,Flourishing Families,12,6,Satellites & Silos,Y1,Young & Wireless,08,6,Savvy Savers,M2,Wealthy Achievers,08,6,06,6,54,6,53,6

Prepare User Options for File Enhancement Batch

Sample UserOptions JSON with All Possible Options

{"headers":[{"cId":"control","fldMeta":"Do Not Import"},{"cId":"std_address","fldMeta":"Address"},{"cId":"std_city","fldMeta":"City"},{"cId":"std_state","fldMeta":"State"},{"cId":"std_zip9","fldMeta":"ZIP"},{"cId":"telephone","fldMeta":"Data"}],"job_details":[{"id":"GEOGRAPHIC_DATA","selections":[{"id":"GEOCODES","options":["STA","STAN","CTY","CTYN","TRA","BGR","CSA","CSAN","CBS","CBSN","DMA","DMAN","PLA","PLAN","MCD","MCDN","TDZ"]}]},{"id":"SEGMENTATION","selections":[{"id":"PZP","selections":[{"id":"include","options":["LifestageGroup","SocialGroup"]},{"id":"geoLevels","options":["ZP6","ZP4","BGR","ZIP"]},{"id":"use","options":["SegmentNames","BestSegment"]}]},{"id":"PSP","selections":[{"id":"include","options":["LifestageGroup"]},{"id":"geoLevels","options":["ZP6"]},{"id":"use","options":["SegmentNames","BestSegment"]}]},{"id":"CNE","selections":[{"id":"include","options":["LifestageGroup"]},{"id":"geoLevels","options":["ZP6"]},{"id":"use","options":["SegmentNames","BestSegment"]}]}]},{"id":"SCORES","selections":[{"id":"IPA","options":["ZP6"]},{"id":"TCHA","options":["ZP6"]},{"id":"TCHU","options":["ZP6"]},{"id":"NTW","options":["ZP6"]}]},{"id":"Output","selections":[{"id":"outputformat","options":["csv"]},{"id":"preferences","options":["AL_CLIENTS"]}]}],"delimiter":"COMMA","matchMode":"best_estimate"}

Sample CSV File

CONTROL	STD_ADDRESS	STD_CITY	STD_STATE	STD_ZIP	TELEPHONE
1	1880 E MAIN ST	PRATTVILLE	AL	36066	3343654555
2	1941A COBBS FORD RD	PRATTVILLE	AL	36066	3343651063

Prepare the Header Section

The Headers section of user options is an array of object of combination of column ID (cId – Column name) and field metadata (fldMeta – System recognisable meta value). The input file has six columns; therefore, the column mapping in the JSON must contain mapping for six columns. Column mapping should be present for all the columns available in the file. To include the column in the output file, it should be categorized as “Data.” If some of the columns are not required, it can be categorized as “Do Not Import.”

FIELD	DESCRIPTION	CID	FLDMETA	REQUIRED
Control	Serial number of the file	control	Do Not Import	Not required.
Std_address	Street address	std_address	Address	Required as per valid combination.
Std_city	City	std_city	City	Required as per valid combination.
Std_state	State	std_state	State	Required as per valid combination.
Std_zip	Zip Code	std_zip	ZIP	Required as per valid combination.

FIELD	DESCRIPTION	CID	FLDMETA	REQUIRED
Telephone	Telephone number of the address.	telephone	Data	Not required. It will be loaded and delivered along with output.

Below are the possible field meta columns and can be recognized as an input column for enhancing a file.

FLDMETA	DESCRIPTION
Address	Address
City	City
State	State
ZIP	ZIP/ZIP+4
Latitude	Latitude of the address
Longitude	Longitude of the address
Block Group	Block group
Usage/Consumption	Usage or consumption value
Do Not Import	The column which is not needed to import
Data	A column which is needed to be retrieved along with output. System won't use this column for any purpose.

The input file must contain any of the following combination of columns to process the file efficiently:

- Address, City, State, ZIP
- Address, City, ZIP
- Address, State, ZIP
- Address, City, State
- Address, ZIP
- Latitude, Longitude
- ZIP
- Block Group

Prepare the Job Details Section

The Job Details section is an array of different mandatory and optional objects such as geographic_data, segmentation, scores, and output. Every object contains ID and SELECTIONS. A selection should include the designated ID and Selections/Options.

- GEOGRAPHIC_DATA (Optional) – This contains geographic variables to append. This is optional. The system must deliver the number of geographies requested.

GEOCODE	DESCRIPTION	OUTPUT COLUMN NAME
STA	State code	STA_GCODE
STAN	State Name	STA_GEOGRAPHIC_NAME
CTY	County code	CTY_GCODE
CTYN	County name	CTY_GEOGRAPHIC_NAME
TRA	Tract code	TRA_GCODE
BGR	Block group code	BGR_GCODE
CSA	Combined Statistical Area Code	CSA_GCODE
CSAN	Combined Statistical Area Name	CSA_GEOGRAPHIC_NAME
CBS	Core Based Statistical Area Code	CBS_GCODE
CBSN	Core Based Statistical Area Name	CBS_GEOGRAPHIC_NAME
DMA	Nielsen Designated Market Area Code	DMA_GCODE
DMAN	Nielsen Designated Market Area Name	DMA_GEOGRAPHIC_NAME
PLA	Place Code	PLA_GCODE
PLAN	Place Name	PLA_GEOGRAPHIC_NAME
MCD	Minor Civil Division Code	MCD_GCODE
MCDN	Minor Civil Division Name	MCD_GEOGRAPHIC_NAME
TDZ	Three Digit ZIP Code	TDZ_GCODE
CNG	Congressional District Code	CNG_GCODE
CNGN	Congressional District Name	CNG_GEOGRAPHIC_NAME

- SEGMENTATION (Optional with data license) - This allows you to specify the required segmentations. It contains an array of selections in the main selection.
- INCLUDE – Adds the alias code and name of the field.
- GEOLEVELS – A mandatory input for each segment. For each geolevel, a set of corresponding output columns will be created.

- USE –
    - SegmentNames: If this parameter is passed, the segment name and include option name (life stage group name/social group name) will be appended.
    - BestSegment: If a segment is not found for the particular input records, the system will go one level up and will look for segment details with the corresponding input by passing this parameter (e.g., if the PZP-ZP6 code segment is not found, the system will use PZP-ZP4 code to get the segment details).

SEGMENTATION	SEGMENTATION SYSTEM NAME	INCLUDE	GEOLEVELS	USE
PZP	PRIZM Premier	LifestageGroup, SocialGroup	ZP6, ZP4, BGR, ZIP	SegmentNames, BestSegment
PSP	P$YCLE Premier	LifestageGroup	ZP6	SegmentNames, BestSegment
CNE	ConneXions	LifestageGroup	ZP6	SegmentNames, BestSegment

Sample Output Column Names

SEGMET-GEOLEVEL COMBINATION	COLUMN NAME
PZP – ZP6	PZP_ZP6_SEG_ID, PZP_ZP6_FLAG, PZP_ZP6_SEG_NAME, PZP_ZP6_LIFESTAGE_GROUP_ALIAS, PZP_ZP6_LIFESTAGE_GROUP_NAME, PZP_ZP6_SOCIAL_GROUP_ALIAS, PZP_ZP6_SOCIAL_GROUP_NAME
PSP – ZP6	PSP_ZP6_SEG_ID, PSP_ZP6_FLAG, PSP_ZP6_SEG_NAME, PSP_ZP6_LIFESTAGE_GROUP_ALIAS, PSP_ZP6_LIFESTAGE_GROUP_NAME
CNE – ZP6	CNE_ZP6_SEG_ID, CNE_ZP6_FLAG, CNE_ZP6_SEG_NAME, CNE_ZP6_LIFESTAGE_GROUP_ALIAS, CNE_ZP6_LIFESTAGE_GROUP_NAME

- SCORES (Optional with data license) – This allows you to specify the required scores. It contains an array of selections.
- GEOLEVELS – A mandatory input for each segment. For each geolevel, a set of corresponding output columns will be created. Below are the available Scores and its corresponding output columns.

SEGMENT-GEOLEVEL COMBINATION	DESCRIPTION	COLUMN NAME
IPA – ZP6	Claritas Income Producing Assets Indicators (IPA) score with ZIP+6 level	IPA_ZP6_SCORE_ID, IPA_ZP6_FLAG
NTW – ZP6	Claritas Net Worth Indicators (NTW) score with ZIP+6 level	NTW_ZP6_SCORE_ID, NTW_ZP6_FLAG
TCHU – ZP6	Claritas Consumer Score – Heavy Technology Usage (TCHU) score with ZIP+6 level	TCHU_ZP6_SCORE_ID, TCHU_ZP6_FLAG
TCHA – ZP6	Claritas Consumer Score – Technology Adoption (TCHA) score with ZIP+6 level	TCHA_ZP6_SCORE_ID, TCHA_ZP6_FLAG

- OUPUT (Mandatory) – The Output section must be added in the **JobDetails** sections, along with the type of output file. You can also mention whether the file and its output need to be stored for future use.
    - **outputformat:** This parameter lets you specify the output format type.
    - **preferences:** By providing this parameter and a name, the file will be stored for future reference.

    Other Parameters:

    - **Delimiter:** This parameter allows you to specify the delimiter of the input or output file.
    - **Input:** {“delimiter”:”COMMA”.}
    - **Output:** { “id”:”OUTPUT”,”selections”:[{ “id”:”outputformat”, “options”:[“txt”]},{ “id”:”delimiter”,”options”:[“PIPE”]}]}
    - **Match Mode:** This parameter provides the mode of matching the addresses in flat files.

Possible Parameter Values:

PARAMETER	DESCRIPTION	POSSIBLE INPUT VALUE(S)
outputformat	Format of the output file	CSV, TXT
preferences	By providing a name for this parameter, the file will be stored for the future.	<user_given_name>
delimiter	Delimiter of the output file.	COMMA, PIPE, TAB SEMI-COLON, CAP
matchMode	Mode of matching the input addresses.	all_possibilities, best_estimate, reject

Default Columns Appended for FE Geocoding

The columns below will be generated in the output files irrespective of user options.

COLUMN NAME	DESCRIPTION
Standardized_address	Standard address for the input address. (This column will not be appended if the input doesn't provide address field / BGR input)
Standardized_city	Standard city for the input address. (This column will not be appended if the input is BGR/ZIP/ZP4 file)
Standardized_state	Standard state for the input address. (This column will not be appended if the input is BGR/ZIP/ZP4 file)
Standardized_zip	Standard ZIP for the input address. (This column will not be appended if the input is BGR/ZIP/ZP4 file)
Latitude	Latitude of the address
Longitude	Longitude of the address
Location_code	Location code
Match_code	Match code
Match_flag	Match flag
Carrier_route_code	Carrier route code
Zip_gcode	ZIP Code
Zp4_gcode	ZIP+4 Code
Zp6_gcode	ZIP+6 Code

Sample Code Snippet for Calling Services

Here is a sample code snippet for calling web services in Java: SampleTestService_java

BUSINESS LIST SERVICES

The Business List services enable you to acquire information about other businesses within geographic areas based on geographic information you specify. Geographic areas can be specified by designating a center point with a radius (or radii), a multi-polygon, or by using standard geographic locations.

Submit Business List Service Call

The Submit Business List service call allows you to send geographic information to the **submitBusinessList** rest call which returns a **jobId** if the call was successful. The jobId can then be used to track the request and is used in other rest calls. The inputs for **submitBusinessList** are specified in JSON which forms the body of the POST rest call.

The table below details the keys and values used in the JSON that forms the body of the submitBusinessList POST rest call.

KEY	VALUE	NOTES
"report_id"	"full_business_list" "basic_business_list" "bpl_basic_analytical_list" "bpl_advanced_analytical_list" "bpl_complete_analytical_list"	
"prompt_id"	"areas"	This prompt_id is used in conjunction with the sub key **geos** (see below).
	"output"	The only available output option is "CSV". This prompt_id can be omitted.
	"nth_record"	If this is set to **10** for example, only every 10th record will be shown. This prompt_id cannot be used in conjunction with **record_limit** below.
	"record_limit"	If this is set to 100 for example, this will limit the total number of returned records to 100. This cannot be used in conjunction with **nth_record** above.
	"report_name"	Set this value to assign a report name. If this is not set, the name will default to **My Business List**. The report name cannot contain special characters.

KEY	VALUE	NOTES
	"filter"	Values here are filters on the NAICS, SIC and Franchise codes such as: (SIC_CODE_1 like '12%') AND (NAICS_CODE_1 like '11%') or complex filters like: (DMA_GCODE IN ('501','502')) AND NOT (CSA_GCODE IN ('108','120')) OR (CSA_GCODE IN ('01002','01004')) OR NOT (CSA_GCODE like 'ABBOTSFORD') **NOTE:** Tables for NAICS, SIC, Franchise and standard geos can be referenced for writing filters (similar to the one shown above): NAICS SIC Franchise Claritas Business-Facts Lists

The **geos** key is associated with three types of geo objects. See the following examples.

KEY	VALUE	GEO JSON OBJECT EXAMPLE	NOTES

"type"	"radius"	{
		"dynamicArea": "true",
		"type": "**radius**",
		"name":"MyRadius1",
		"latLongs": [{
		"latitude": "39.226557",
		"longitude": "-84.353822"
		}],
		"radii": ["1", "3", "5"]
		}
			Latlongs (latitude and longitude) specifies the center of the circle.
			The units of the radii are in miles.

	"multipolygon"	{
		"dynamicArea": "true",
		"type": "**multipolygon**",
		"analysisAreaName":"MyPolygon1",
		"wkt": "MULTIPOLYGON((
		(-76.521898408 39.051802928,
		-76.516877775 39.050647885,
		-76.504069092 39.041499947)))"
		}
			Multi-polygons are specified by a set of coordinates that correspond to any number of polygons. These polygons are each defined by latlong points that define a particular polygon's area.

	(type not set)
	Standard GEO
		{
		"level_codes": ["01"],
		"geo_level": "STA",
		"level_names": ["Alabama"]
		}
			Standard geos are specified by level codes and geo levels.
			Geo_Levels

Below is a link to a JSON example that shows how you can use the JSON keys and values detailed in previous tables to construct JSON for the body of the **submitBusinessList** rest call.

Sample JSON Files

business_list

multiple_zips

business_list_complex_filter

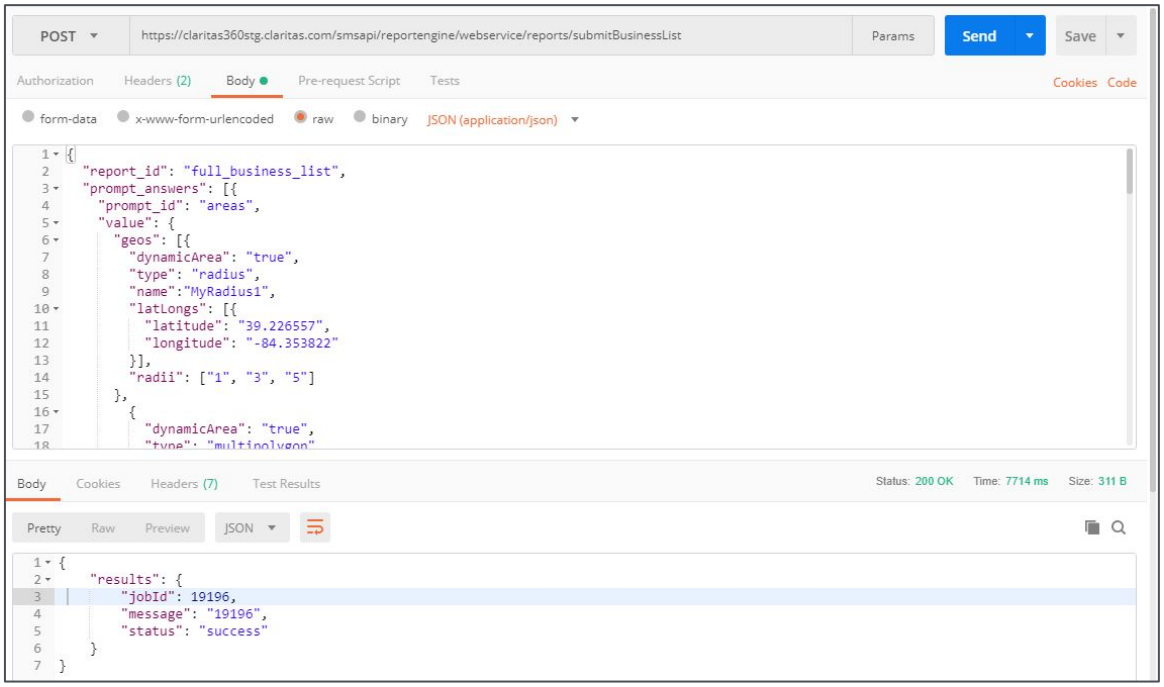
complex_filter

Test submitBusinessList Call in Postman

To test the **submitBusinessList** call in Postman, perform the following:

1. Open the Postman application.
2. Generate an access token. Refer to the "Authentication and Authorization to Use Web Services – Authentication" section for instructions on page 5.
3. Run the following **Post** method on the following URL:
4. https://claritas360.claritas.com/smsapi/reportengine/webservice/reports/submitBusinessList
5. For Testing: https://claritas360stg.claritas.com/smsapi/reportengine/webservice/reports/submitBusinessList
6. Select **Headers**, add **access_token** under **Key**, and then copy the **access_token** output from the **Authentication call** to **Value**.
7. Add your JSON specifying your inputs (for ex. use contents of **business_list.json** above) to the Body section of Postman as raw JSON (application/json)

8. Click **Send**.



Screen showing how to test the **submitBusinessList** call

This results in JSON output containing a jobId if the rest call is successful.

Response

response => {"results":{"jobId":28110,"message":"28110","status":"success"}}

Sample Java Code Using Spring's RestTemplate to Call submitBusinessList

```
ResponseEntity<String> submitBusinessList(String accessToken) throws IOException {
   RestTemplate restTemplate = new RestTemplate();

   HttpHeaders headers = new HttpHeaders();
   headers.setContentType(MediaType.APPLICATION_JSON);
   headers.set("access_token", accessToken);

   String resourceUrl =
"https://claritas360.claritas.com/smsapi/reportengine/webservice/reports/submitBusinessList";

   ClassPathResource jsonResource = new ClassPathResource("business_list.json");
   String jsonBody = FileUtils.readFileToString(jsonResource.getFile(),(String)null);

   HttpEntity<String> entity = new HttpEntity<>(jsonBody, headers);
   return restTemplate.exchange(resourceUrl, HttpMethod.POST, entity, String.class);
}
```

For Testing

```
ResponseEntity<String> submitBusinessList(String accessToken) throws IOException {
    RestTemplate restTemplate = new RestTemplate();

    HttpHeaders headers = new HttpHeaders();
    headers.setContentType(MediaType.APPLICATION_JSON);
    headers.set("access_token", accessToken);

    String resourceUrl =
"https://claritas360stg.claritas.com/smsapi/reportengine/webservice/reports/submitBusinessList";

    ClassPathResource jsonResource = new ClassPathResource("business_list.json");
    String jsonBody = FileUtils.readFileToString(jsonResource.getFile(),(String)null);

    HttpEntity<String> entity = new HttpEntity<>(jsonBody, headers);
    return restTemplate.exchange(resourceUrl, HttpMethod.POST, entity, String.class);
}
```

In the previous code snippet, the accessToken returned by the previous **getAccessToken()** call is used as the argument to the method call of **submitBusinessList()** above. The file **business_list.json** (see the contents of the JSON file on page 5) is pulled from the resources folder of the java project. The class **ClassPathResource** is used to retrieve the **business_list.json** file from the resources folder, which is passed to the body argument of the **RestTemplate** call.

Response

```
response =>  <200,{"results":{"jobId":28110,"message":"28110","status":"success"}} ...>
```

Submit Business Count Service Call

The Submit Business Count service call allows you to obtain the anticipated number of results before actually making a call to *submitBusinessList*. This is a synchronous call that returns results immediately in the response. The inputs for **submitBusinessCount** are the same as *submitBusinessList* (see above).

Test submitBusinessCount Call in Postman

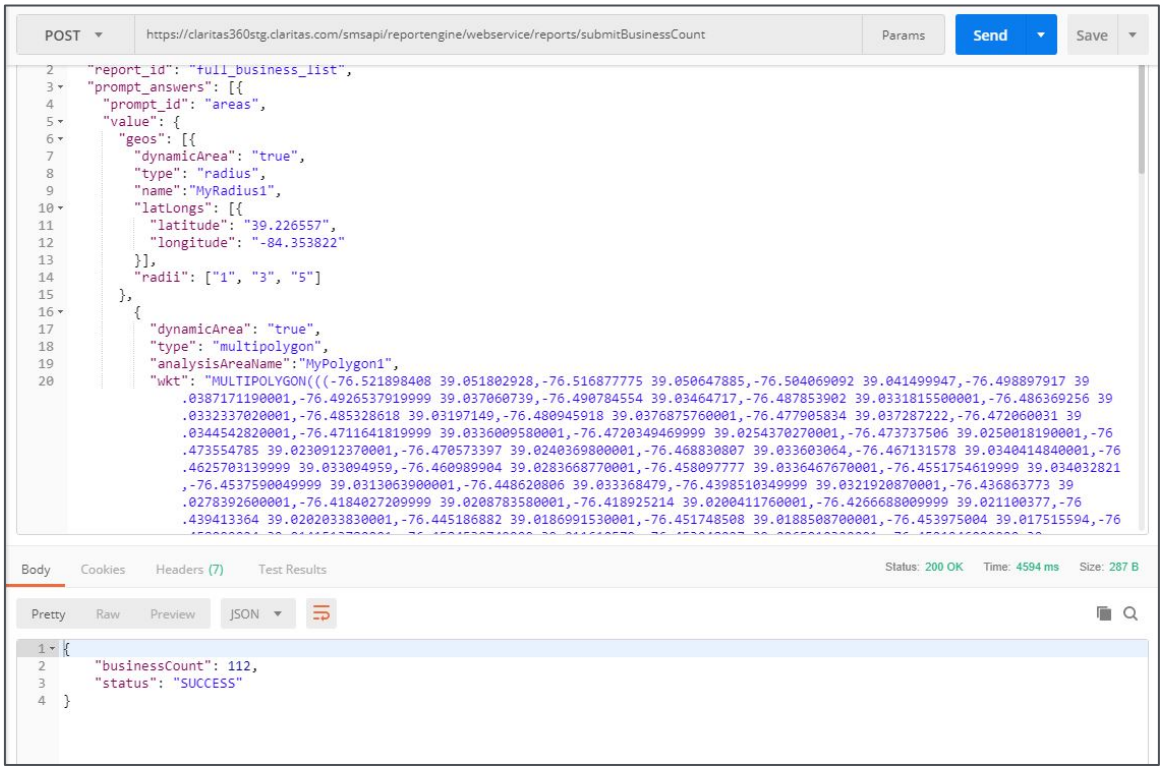
To test the **submitBusinessCount** call in Postman, perform the following:

1. Open the Postman application.
2. Generate an access token. Refer to the “Authentication and Authorization to Use Web Services – Authentication” section for instructions on page 5.
3. Run the following **Post** method on the following URL:
    - https://claritas360.claritas.com/smsapi/reportengine/webservice/reports/submitBusinessCount
    - For Testing: https://claritas360stg.claritas.com/smsapi/reportengine/webservice/reports/submitBusinessCount

4. Select **Headers**, add **access_token** under **Key**, and then copy the **access_token** output from the **Authentication call** to **Value**.
5. Add your JSON specifying your inputs (for ex. use contents of **business_list.json** above) to the Body section of Postman as raw JSON (application/json)
6. Click **Send**.

Response

response => {"businessCount":112,"status":"SUCCESS"}



Screen showing how to test the **submitBusinessCount** call

Get Job Status Service Call

The Get Job Status Service call will return the report of the requested business list using the previously returned jobId if the requested processing is complete. If the previously submitted business list request has not yet completed getJobStatus will return a response indicating that the request has not yet been completed.

Calling getJobStatus Using Postman to return Job Status

1. Run the **GET** method on the following URL using the previously returned jobId:
2. https://claritas360.claritas.com/smsapi/reportengine/webservice/reports/getJobStatus?jobId=<jobId from last call>&downloadResults=false

3. For Testing: https://claritas360stg.claritas.com/smsapi/reportengine/webservice/reports/getJobStatus?jobId=<jobId from last call>&downloadResults=false
4. Select **Headers**, add **access_token** under **Key**, and then copy the **access_token** output from the **Authentication call** to **Value.**
5. Click **Send.** The Status of the Business List job will be returned (Examples Below).
6. { "id": "19196", "jobStatus": "NOT_STARTED"}
7. { "id": "19196", "jobStatus": "IN_PROGRESS"}
8. { "id": "19196", "jobStatus": "COMPLETED"}

Screen showing a returned business list Job Status

Sample Java Code to Call getJobStatus Using Spring's RestTemplate to return Job Status

```
ResponseEntity<String> getJobStatus(String accessToken, Integer jobId) {
   RestTemplate restTemplate = new RestTemplate();

   HttpHeaders headers = new HttpHeaders();
   headers.setContentType(MediaType.APPLICATION_JSON);
   headers.set("access_token", accessToken);
   HttpEntity<String> entity = new HttpEntity<>("parameters", headers);

   String resourceUrl =
"https://claritas360.claritas.com/smsapi/reportengine/webservice/reports/getJobStatus";

   UriComponentsBuilder builder = UriComponentsBuilder.fromUriString(resourceUrl)
         .queryParam("downloadResults", false)
         .queryParam("job_id", JobId);

   return restTemplate.exchange(builder.toUriString(), HttpMethod.GET, entity, String.class);
}
```

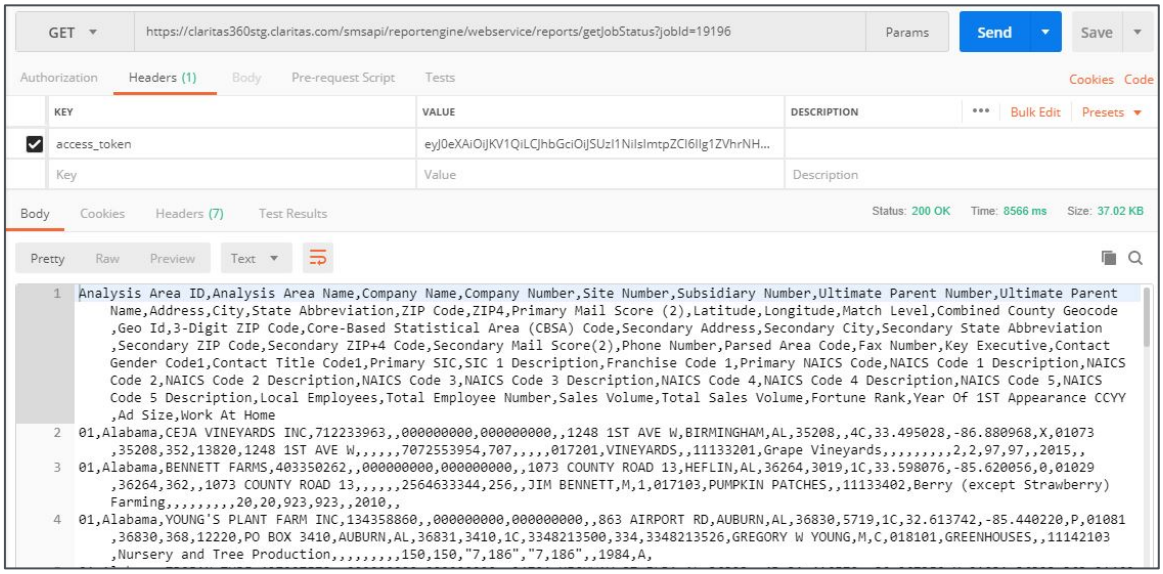
For Testing

```
ResponseEntity<String> getJobStatus(String accessToken, Integer jobId) {
   RestTemplate restTemplate = new RestTemplate();

   HttpHeaders headers = new HttpHeaders();
   headers.setContentType(MediaType.APPLICATION_JSON);
```

```
    headers.set("access_token", accessToken);
    HttpEntity<String> entity = new HttpEntity<>("parameters", headers);

    String resourceUrl =
"https://claritas360stg.claritas.com/smsapi/reportengine/webservice/reports/getJobStatus";

    UriComponentsBuilder builder = UriComponentsBuilder.fromUriString(resourceUrl)
            .queryParam("downloadResults", false)
           .queryParam("job_id", JobId);

    return restTemplate.exchange(builder.toUriString(), HttpMethod.GET, entity, String.class);
}
```

Calling getJobStatus Using Postman to return Business List Report

1. Run the **GET** method on the following URL using the previously returned jobId:
   - https://claritas360.claritas.com/smsapi/reportengine/webservice/reports/getJobStatus?jobId=<jobId from last call>
   - For Testing: https://claritas360stg.claritas.com/smsapi/reportengine/webservice/reports/getJobStatus?jobId=<jobId from last call>
2. Select **Headers**, add **access_token** under **Key**, and then copy the **access_token** output from the **Authentication call** to **Value.**
3. Click **Send.** The Status of the Business List Return will be returned.



Sample Java Code to Call getJobStatus Using Spring's RestTemplate to return the Report

```
ResponseEntity<String> getJobStatus(String accessToken, Integer jobId) {
    RestTemplate restTemplate = new RestTemplate();
```

```
    HttpHeaders headers = new HttpHeaders();
    headers.setContentType(MediaType.APPLICATION_JSON);
    headers.set("access_token", accessToken);
    HttpEntity<String> entity = new HttpEntity<>("parameters", headers);

    String resourceUrl =
"https://claritas360.claritas.com/smsapi/reportengine/webservice/reports/getJobStatus";

    UriComponentsBuilder builder = UriComponentsBuilder.fromUriString(resourceUrl)
            .queryParam("jobId", jobId);

    return restTemplate.exchange(builder.toUriString(), HttpMethod.GET, entity, String.class);
}
```

For Testing

```
ResponseEntity<String> getJobStatus(String accessToken, Integer jobId) {
    RestTemplate restTemplate = new RestTemplate();

    HttpHeaders headers = new HttpHeaders();
    headers.setContentType(MediaType.APPLICATION_JSON);
    headers.set("access_token", accessToken);
    HttpEntity<String> entity = new HttpEntity<>("parameters", headers);

    String resourceUrl =
"https://claritas360stg.claritas.com/smsapi/reportengine/webservice/reports/getJobStatus";

    UriComponentsBuilder builder = UriComponentsBuilder.fromUriString(resourceUrl)
            .queryParam("jobId", jobId);

    return restTemplate.exchange(builder.toUriString(), HttpMethod.GET, entity, String.class);
}
```

In the previous code snippet, the accessToken returned by the previous **getAccessToken()** call is used as the first argument to the method call of **getJobStatus()** above. The jobId returned by the previous **submitBusinessList()** call is used as the second argument to the method call of getJobStatus() above. If the business list request process has been completed a business list will be returned in the returned ResponseEntity (see below).

Response

```
<200,My Business List,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
Analysis Area ID,Analysis Area Name,Company Name,Company Number,Site Number,Subsidiary
Number,Subsidiary Name,Ultimate Parent Number,Ultimate Parent Name,Address,City,State
Abbreviation,ZIP Code,ZIP4,Primary Mail Score (2),Latitude,Longitude,Match Level,Combined County
Geocode,ZIP Code,3-Digit ZIP Code,Core-Based Statistical Area (CBSA) Code,Secondary
Address,Secondary City,Secondary State Abbreviation,Secondary ZIP Code,Secondary ZIP+4
Code,Secondary Mail Score(2),Phone Number,Parsed Area Code,Fax Number,Key

. . .
```

Additional Reference Tables

GEOGRAPHIC_DATA – This contains geographic variables used by the standard geo JSON objects.

GEOCODE	DESCRIPTION	OUTPUT COLUMN NAME
STA	State code	STA_GCODE
STAN	State Name	STA_GEOGRAPHIC_NAME
CTY	County code	CTY_GCODE
CTYN	County name	CTY_GEOGRAPHIC_NAME
TRA	Tract code	TRA_GCODE
BGR	Block group code	BGR_GCODE
CSA	Combined Statistical Area Code	CSA_GCODE
CSAN	Combined Statistical Area Name	CSA_GEOGRAPHIC_NAME
CBS	Core Based Statistical Area Code	CBS_GCODE
CBSN	Core Based Statistical Area Name	CBS_GEOGRAPHIC_NAME
DMA	Nielsen Designated Market Area Code	DMA_GCODE
DMAN	Nielsen Designated Market Area Name	DMA_GEOGRAPHIC_NAME
PLA	Place Code	PLA_GCODE
PLAN	Place Name	PLA_GEOGRAPHIC_NAME
MCD	Minor Civil Division Code	MCD_GCODE
MCDN	Minor Civil Division Name	MCD_GEOGRAPHIC_NAME
TDZ	Three Digit ZIP Code	TDZ_GCODE
CNG	Congressional District Code	CNG_GCODE
CNGN	Congressional District Name	CNG_GEOGRAPHIC_NAME

Sample Java Code for Calling Business List Services

Attached below are the following: a java file and a pom.xml file that combines all the Java® code snippets referenced throughout the document. You can generate the structure of the project using Spring Boot's Spring Initializer and make use of the attached files to test the examples.

DemoApplication

pom

REPORT SERVICES

The Report services enable you to run reports within geographic areas based on geographic information you specify. Geographic areas can be specified by designating a center point with a radius (or radii), a multi-polygon, or by using standard geographic locations.

Submit Report Service Call

The Submit Report service call allows you to send geographic information to the **submitReportJob** rest call which returns a **jobId** if the call was successful. The jobId can then be used to track the request and is used in other rest calls. The inputs for **submitReportJob** are specified in JSON which forms the body of the POST rest call.

The table below details the keys and values used in the JSON that forms the body of the submitReportJob POST rest call. If the wrong JSON is submitted, the service will return a 404 error.

KEY	VALUE	NOTES
"report_id"	"business_facts_summary" "consumer_buying_power_2018" "pop_facts_demographic_trend" "pop_facts_demographics" "retail_market_power_2018" "segment_distribution" "senior_life" "effective_buying_income" "pop_facts_demographics_by_age_race_sex" "pop_facts_household_income_by_age_of_householder" "pop_facts_executive_summary"	
"prompt_id"	"areas"	This prompt_id is used in conjunction with the sub key **geos** (see below).
	"level_detail"	Any ONE of the following: AS_SELECTED AGGREGATE COMPONENT AGGREGATE_AND_COMPONENT
	"include_chart"	Any ONE of the following: true false Not available for: "segment_distribution" "retail_market_power_2018"

KEY	VALUE	NOTES
		"consumer_buying_power_2018" "business_facts_summary"
	"include_map"	Any ONE of the following: true false Not available for: "segment_distribution"
	"color_scheme"	Any ONE of the following: "claritas-standard" "office" "hotcold" "sunset" "orange" "blue" "green" "grape" "grey" Only available when include_map is true for "effective_buying_income" "pop_facts_demographics_by_age_race_sex" "pop_facts_household_income_by_age_of_householder" "pop_facts_executive_summary"
	"map_labels"	Any ONE of the following: true false Only available when include_map is true for "effective_buying_income" "pop_facts_demographics_by_age_race_sex" "pop_facts_household_income_by_age_of_householder" "pop_facts_executive_summary"
	"map_subtotal_method"	Any ONE of the following: NONE EQUAL_RANGE EQUAL_GEO NTILE Only available when include_map is true for

KEY	VALUE	NOTES
		"effective_buying_income" "pop_facts_demographics_by_age_race_sex" "pop_facts_household_income_by_age_of_householder" "pop_facts_executive_summary"
	theme_variable	Only available when include_map is true AND map_subtotal_type is NOT none for "effective_buying_income" "pop_facts_demographics_by_age_race_sex" "pop_facts_household_income_by_age_of_householder" "pop_facts_executive_summary"
	theme_sort_measure	Any ONE of the following: INDEX BASE_COUNT BASE_COMP COUNT PER_COMP PER_PEN Only available when include_map is true AND map_subtotal_type is NOT none for "effective_buying_income" "pop_facts_demographics_by_age_race_sex" "pop_facts_household_income_by_age_of_householder" "pop_facts_executive_summary"
	theme_sort_direction	Any ONE of the following: ASC DESC Only available when include_map is true AND map_subtotal_type is NOT none for senior_life effective_buying_income pop_facts_executive_summary pop_facts_household_income_by_age_of_householder pop_facts_demographics_by_age_race_sex pop_facts_demographic_trend pop_facts_demographics segment_distribution business_facts_summary

KEY	VALUE	NOTES
		consumer_buying_power_2018 retail_market_power_2018
	theme_number_ranges	Any ONE of the following: 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 Only available when include_map is true AND map_subtotal_type is NOT none for "effective_buying_income" "pop_facts_demographics_by_age_race_sex" "pop_facts_household_income_by_age_of_householder" "pop_facts_executive_summary"
	theme_subtotal_measure	Any ONE of the following: COUNT BASE_COUNT Only available when include_map is true AND map_subtotal_type is NTILE for "effective_buying_income" "pop_facts_demographics_by_age_race_sex" "pop_facts_household_income_by_age_of_householder" "pop_facts_executive_summary"
	"generation_method"	Any ONE of the following: SINGLE ONE_PER_AREA Not available for: "segment_distribution"

KEY	VALUE	NOTES
	"output_type"	Any ONE of the following: CSV EXCEL PDF Not available for: "segment_distribution"
	"email"	Any ONE of the following: DO_NOT_SEND SEND_NOTIFICATION SEND_FILES
	"report_name"	Set this value to assign a report name. The report name cannot contain special characters.
	"sections_business_facts_summary"	Any combination of: 3_digit_naics_summary construction_and_manufacturing healthcare_naics_summary retail_naics_summary business_summary_occupation services_naics_summary workplace_business_and_employment workplace_bus_and_emp_summary 3_digit_naics_top_ten_summary ONLY available for: "business_facts_summary"
	"sections_consumer_buying_power_2018"	Any combination of: summary food alcoholic_beverages housing apparel_and_services transportation healthcare entertainment personal_care read_edu_tobacco exp_contr_ins spending_patterns retail_store_types ONLY available for: "consumer_buying_power_2018"

KEY	VALUE	NOTES
	"sections_pop_facts_demographic_trend"	Any combination of: pop_facts_demographic_trend population_by_age_and_sex_trend household_trend population_summary household_trend_summary ONLY available for: "pop_facts_demographic_trend"
	"sections_pop_facts_demographics"	Any combination of: pop_facts_demographic_snapshot pop_facts_census_demographic_overview pop_facts_population_quick_facts pop_facts_household_quick_facts pop_facts_demographic_quick_facts pop_facts_summary ONLY available for: "pop_facts_demographics"
	"sections_retail_market_power_2018"	Any combination of: oppportunity_gap_merch_lines demand_growth_merch_lines opportunity_gap_retail_stores demand_growth_retail_stores ONLY available for: "retail_market_power_2018"
	"base_analysis_type"	Any ONE of: TOTAL_US PARENT_GEOGRAPHY (OPENS PROMPT parent_geography_level) GEOGRAPHY (OPENS PROMPT base_analysis_area) ONLY available for: "consumer_buying_power_2018" "segment_distribution" "pop_facts_executive_summary"
	"parent_geography_level"	Any ONE of: USA STA CSA DMA DMA TDZ

KEY	VALUE	NOTES
		CBS CTY PLA ZIP MCD TRA BGR CBL CBM CBN MTA LAT BTA RSA LEC CTR CNG ONLY available for: "segment_distribution" "consumer_buying_power_2018" "pop_facts_executive_summary"
	"base_analysis_area"	This prompt_id is used in conjunction with the sub key **geos** (see below). ONLY available for: "segment_distribution" "pop_facts_executive_summary"
	"segment_system"	Any ONE of: PZP PSP CNE ONLY available for: "segment_distribution"
	"projection_dataset"	Any combination of: CY FY ZP6_CY ONLY available for: "segment_distribution"
	"sort_segment_code"	Any ONE of: ROW_ID SELECTION

KEY	VALUE	NOTES
		ONLY available for: "segment_distribution"
	"sort_direction"	Any ONE of: ASC DESC ONLY available for: "segment_distribution"
	"segment_description"	Any ONE of: false true ONLY available for: "segment_distribution"
	"segment_output_type"	Options: EXCEL PDF ONLY available for: "segment_distribution"
	"segment_descriptors"	Enter the appropriate year (replace YYYY with the currently available years) in the below segment descriptors, depending on the data vintage you want used in your report. When segment_system = PZP then any combination of: SEG_YYYY_PZP_HH_AGE_RANGE SEG_YYYY_PZP_HH_COMPOSITION SEG_YYYY_PZP_HH_EDUCATION SEG_YYYY_PZP_HH_EMPLOYMENT SEG_YYYY_PZP_HH_IPA_CLASS SEG_YYYY_PZP_HH_INCOME SEG_YYYY_PZP_HH_TENURE SEG_YYYY_PZP_LIFESTAGE_GROUP_NAME SEG_YYYY_PZP_SOCIAL_GROUP_NAME SEG_YYYY_PZP_URBANICITY When segment_system = PSP then any combination of: SEG_YYYY_PSP_HH_AGE_RANGE SEG_YYYY_PSP_HH_COMPOSITION SEG_YYYY_PSP_HH_EDUCATION SEG_YYYY_PSP_HH_EMPLOYMENT

KEY	VALUE	NOTES
		SEG_YYYY_PSP_HH_IPA_CLASS SEG_YYYY_PSP_HH_INCOME SEG_YYYY_PSP_HH_RACE_AND_ETHNICITY_RANGE SEG_YYYY_PSP_HH_TENURE SEG_YYYY_PSP_LIFESTAGE_GROUP_NAME SEG_YYYY_PSP_URBANICITY When segment_system = CNE then any combination of: SEG_YYYY_CNE_HH_AGE_RANGE SEG_YYYY_CNE_HH_COMPOSITION SEG_YYYY_CNE_HH_EDUCATION SEG_YYYY_CNE_HH_EMPLOYMENT SEG_YYYY_CNE_HH_INCOME SEG_YYYY_CNE_HH_RACE_AND_ETHNICITY_RANGE, SEG_YYYY_CNE_HH_TENURE SEG_YYYY_CNE_LIFESTAGE_GROUP_NAME SEG_YYYY_CNE_TECH_CLASS SEG_YYYY_CNE_URBANICITY ONLY available for: "segment_distribution"

The **geos** key is associated with three types of geo objects. See the following examples.

KEY	VALUE	GEO JSON OBJECT EXAMPLE	NOTES

"type"	"radius"	{
		"dynamicArea": "true",
		"type": "**radius**",
		"name":"MyRadius1",
		"latLongs": [{
		"latitude": "39.226557",
		"longitude": "-84.353822"
		}],
		"radii": ["1"]
		},
		{
		"dynamicArea": "true",
		"type": "**radius**",
		"name":"MyRadius3",
		"latLongs": [{
		"latitude": "39.226557",
		"longitude": "-84.353822"
		}],
			Latlongs (latitude and longitude) specifies the center of the circle.
			The units of the radii are in miles.

KEY	VALUE	GEO JSON OBJECT EXAMPLE	NOTES

		"radii": ["3"]
		},
		{
		"dynamicArea": "true",
		"type": "**radius**",
		"name":"MyRadius5",
		"latLongs": [{
		"latitude": "39.226557",
		"longitude": "-84.353822"
		}],
		"radii": ["5"]
		}	

	"multipolygon"	{
		"dynamicArea": "true",
		"type": "**multipolygon**",
		"analysisAreaName":"MyPolygon1",
		"wkt": "MULTIPOLYGON((
		(-76.521898408 39.051802928,
		-76.516877775 39.050647885,
		-76.504069092 39.041499947)))"
		}
			Multi-polygons are specified by a set of coordinates that correspond to any number of polygons. These polygons are each defined by latlong points that define a particular polygon's area.

	(type not set)
	Standard GEO
		{
		"level_codes": ["01"],
		"geo_level": "STA",
		"level_names": ["Alabama"]
		}
			Standard geos are specified by level codes and geo levels.
			Geo_Levels

Here is a link to a JSON example that shows how you can use the JSON keys and values detailed in previous tables to construct JSON for the body of the **submitReportJob** rest call: report.JSON

Sample JSON Files

Business-Facts Summary - Workplace Business and Employment Section

Consumer Buying Power - Consumer Spending Patterns Section (2018)

Effective Buying Income

Pop-Facts Demographic Trend

Pop-Facts Demographics by Age Race Sex

Pop-Facts Demographics - Senior Life

Pop-Facts Executive Summary

Pop-Facts Household Income by Age of Householder

Pop-Facts Household Quick Facts

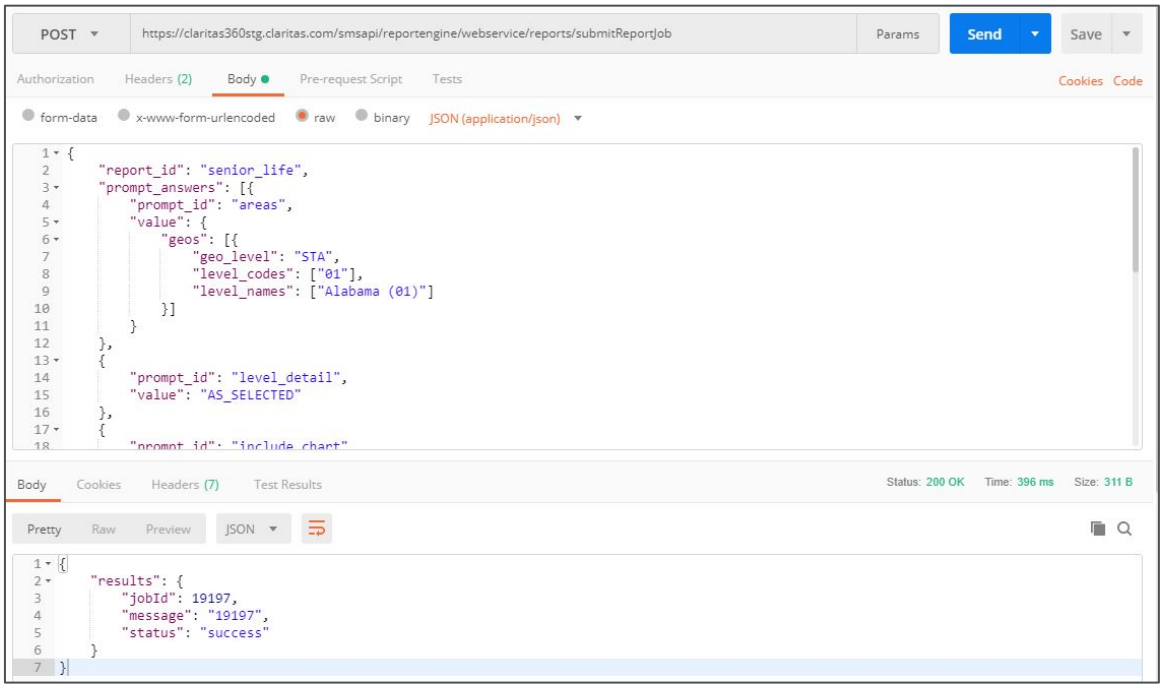
Retail Market Power - Retail Stores Opportunity Section (2018)

Segmentation Distribution

Test submitReportJob Call in Postman

To test the **submitReportJob** call in Postman, perform the following:

1. Open the Postman application.
2. Generate an access token. Refer to the "Authentication and Authorization to Use Web Services – Authentication" section for instructions on page 5.
3. Run the following **Post** method on the following URL:
   - https://claritas360.claritas.com/smsapi/reportengine/webservice/reports/submitReportJob
   - For Testing: https://claritas360stg.claritas.com/smsapi/reportengine/webservice/reports/submitReportJob
4. Select **Headers**, add **access_token** under **Key**, and then copy the **access_token** output from the **Authentication call** to **Value**.
5. Add your JSON specifying your inputs (for ex. use contents of **report.json** above) to the Body section of Postman as raw JSON (application/json).
6. Click on **Send to** submit the report.



Screen showing how to test the **submitReportJob** call

This results in JSON output containing a jobId if the rest call is successful.

Response

```
response => {"results":{"jobId":19197,"message":"19197","status":"success"}}
```

Sample Java Code Using Spring's RestTemplate to Call submitReportJob

```
ResponseEntity<String> submitReportJob(String accessToken) throws IOException {
   RestTemplate restTemplate = new RestTemplate();

   HttpHeaders headers = new HttpHeaders();
   headers.setContentType(MediaType.APPLICATION_JSON);
   headers.set("access_token", accessToken);

   String resourceUrl =
"https://claritas360.claritas.com/smsapi/reportengine/webservice/reports/submitReportJob";

   ClassPathResource jsonResource = new ClassPathResource("business_facts.json");
   String jsonBody = FileUtils.readFileToString(jsonResource.getFile(),(String)null);

   HttpEntity<String> entity = new HttpEntity<>(jsonBody, headers);
   return restTemplate.exchange(resourceUrl, HttpMethod.POST, entity, String.class);
}
```

For Testing

```
ResponseEntity<String> submitReportJob(String accessToken) throws IOException {
   RestTemplate restTemplate = new RestTemplate();

   HttpHeaders headers = new HttpHeaders();
   headers.setContentType(MediaType.APPLICATION_JSON);
   headers.set("access_token", accessToken);

   String resourceUrl =
"https://claritas360.claritas.com/smsapi/reportengine/webservice/reports/submitReportJob";

   ClassPathResource jsonResource = new ClassPathResource("business_facts.json");
   String jsonBody = FileUtils.readFileToString(jsonResource.getFile(),(String)null);

   HttpEntity<String> entity = new HttpEntity<>(jsonBody, headers);
   return restTemplate.exchange(resourceUrl, HttpMethod.POST, entity, String.class);
}
```

In the previous code snippet, the accessToken returned by the previous **getAccessToken()** call is used as the argument to the method call of **submitReportJob()** above. The file **business_facts.json** (see the contents of the JSON file on starting on page 33) is pulled from the resources folder of the java project. The class **ClassPathResource** is used to retrieve the **business_facts.json** file from the resources folder, which is passed to the body argument of the **RestTemplate** call.

Response

```
response =>  <200,{"results":{"jobId":85809,"message":"85809","status":"success"}} ...>
```

Get Job Status Service Call

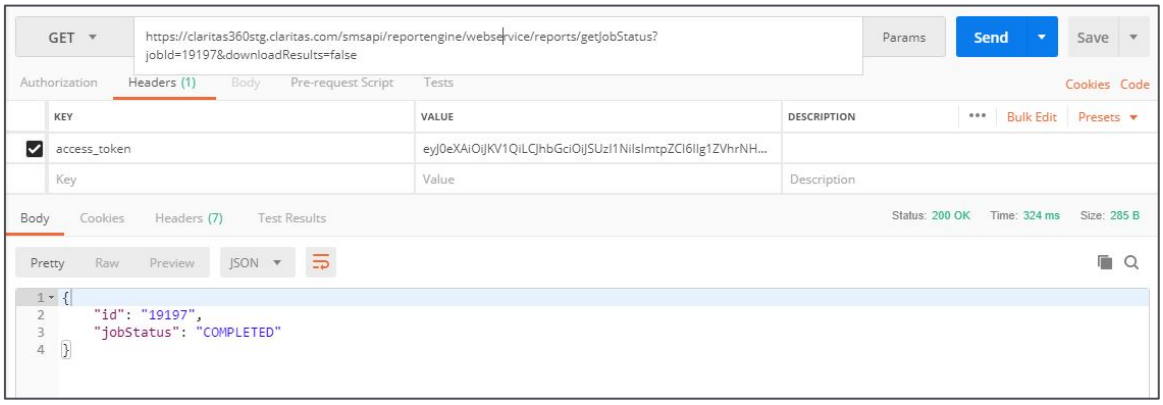
The Get Job Status Service call will return the report of the requested business list using the previously returned jobId if the requested processing is complete. If the previously submitted report request has not yet completed getJobStatus will return a response indicating that the request has not yet been completed.

Special Note about streaming report results in Postman

Report results are streamed back to the user once the job is complete. When doing this in Postman, the results will be seen in the postman window, resulting in results which is not usable. Postman provides a method to capture this into a file as part of a download. The instructions below represent that methodology.

Calling getJobStatus Using Postman to return Job Status

1. Run the **GET** method on the following URL using the previously returned jobId:
   - https://claritas360.claritas.com/smsapi/reportengine/webservice/reports/getJobStatus?jobId=<jobId from last call>&downloadResults=false
   - For Testing: https://claritas360stg.claritas.com/smsapi/reportengine/webservice/reports/getJobStatus?jobId=<jobId from last call>&downloadResults=false
2. Select **Headers**, add **access_token** under **Key**, and then copy the **access_token** output from the **Authentication call** to **Value.**
3. Click **Send.** The Status of the Business List job will be returned (Examples Below).
   - { "id": "19197", "jobStatus": "NOT_STARTED"}
   - { "id": "19197", "jobStatus": "IN_PROGRESS"}
   - { "id": "19197", "jobStatus": "COMPLETED"}



Screen showing a returned report

Sample Java Code to Call getJobStatus Using Spring's RestTemplate to Return Job Status

```
 ResponseEntity<String> getJobStatus(String accessToken, Integer jobId) {
    RestTemplate restTemplate = new RestTemplate();

    HttpHeaders headers = new HttpHeaders();
    headers.setContentType(MediaType.APPLICATION_JSON);
    headers.set("access_token", accessToken);
    HttpEntity<String> entity = new HttpEntity<>("parameters", headers);

    String resourceUrl =
"https://claritas360.claritas.com/smsapi/reportengine/webservice/reports/getJobStatus";

    UriComponentsBuilder builder = UriComponentsBuilder.fromUriString(resourceUrl)
.queryParam("downloadResults", false)
.queryParam("jobId", jobId);

    return restTemplate.exchange(builder.toUriString(), HttpMethod.GET, entity, String.class);
}
```

For Testing

```
ResponseEntity<String> getJobStatus(String accessToken, Integer jobId) {
    RestTemplate restTemplate = new RestTemplate();

    HttpHeaders headers = new HttpHeaders();
    headers.setContentType(MediaType.APPLICATION_JSON);
    headers.set("access_token", accessToken);
    HttpEntity<String> entity = new HttpEntity<>("parameters", headers);

    String resourceUrl =
"https://claritas360stg.claritas.com/smsapi/reportengine/webservice/reports/getJobStatus";

    UriComponentsBuilder builder = UriComponentsBuilder.fromUriString(resourceUrl)
.queryParam("downloadResults", false)
.queryParam("jobId", jobId);

    return restTemplate.exchange(builder.toUriString(), HttpMethod.GET, entity, String.class);
}
```

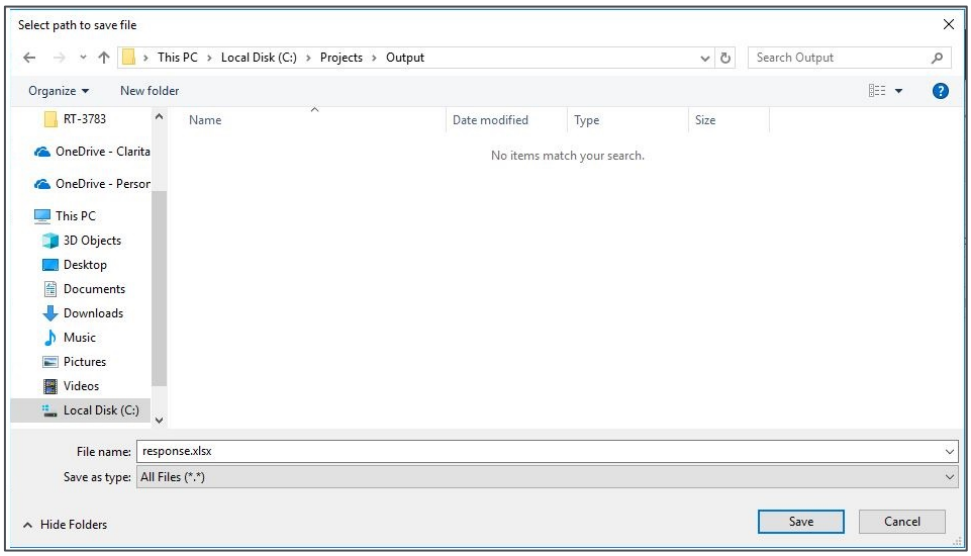
Calling getJobStatus Using Postman to return the Report

1. Run the **GET** method on the following URL using the previously returned jobId:
    - https://claritas360.claritas.com/smsapi/reportengine/webservice/reports/getJobStatus?jobId=<jobId from last call>
    - For Testing: https://claritas360stg.claritas.com/smsapi/reportengine/webservice/reports/getJobStatus?jobId=<jobId from last call>
2. Select **Headers**, add **access_token** under **Key**, and then copy the **access_token** output from the **Authentication call** to **Value.**

3. In Postman, to save the report, click on the arrow next to **Send** and select **Send and Download.**

Screen showing a retrieving a report using Send and Download

4. You will be prompted to save the file on your local file system.



Screen showing saving the file through postman

5. Name the file with the appropriate File Extension:
   - .csv for Comma Separated
   - .pdf for PDF files
   - .xlsx for Excel Files

Sample Java Code to Call getJobStatus Using Spring's RestTemplate to Return Report

```
ResponseEntity<String> getJobStatus(String accessToken, Integer jobId) {
  RestTemplate restTemplate = new RestTemplate();

  HttpHeaders headers = new HttpHeaders();
  headers.setContentType(MediaType.APPLICATION_JSON);
  headers.set("access_token", accessToken);
  HttpEntity<String> entity = new HttpEntity<>("parameters", headers);

  String resourceUrl =
"https://claritas360.claritas.com/smsapi/reportengine/webservice/reports/getJobStatus";

  UriComponentsBuilder builder = UriComponentsBuilder.fromUriString(resourceUrl)
```

```
        .queryParam("jobId", jobId);

    return restTemplate.exchange(builder.toUriString(), HttpMethod.GET, entity, String.class);
}
```

For Testing

```
ResponseEntity<String> getJobStatus(String accessToken, Integer jobId) {
    RestTemplate restTemplate = new RestTemplate();

    HttpHeaders headers = new HttpHeaders();
    headers.setContentType(MediaType.APPLICATION_JSON);
    headers.set("access_token", accessToken);
    HttpEntity<String> entity = new HttpEntity<>("parameters", headers);

    String resourceUrl =
"https://claritas360stg.claritas.com/smsapi/reportengine/webservice/reports/getJobStatus";

    UriComponentsBuilder builder = UriComponentsBuilder.fromUriString(resourceUrl)
        .queryParam("jobId", jobId);

    return restTemplate.exchange(builder.toUriString(), HttpMethod.GET, entity, String.class);
}
```

In the previous code snippet, the accessToken returned by the previous **getAccessToken()** call is used as the first argument to the method call of **getJobStatus()** above. The jobId returned by the previous **submitReportJob()** call is used as the second argument to the method call of getJobStatus() above. If the business list request process has been completed a business list will be returned in the returned ResponseEntity (see below).

Response

<200> and the file will be streamed upon completion

Additional Reference Tables

GEOGRAPHIC_DATA – This contains geographic variables used by the standard geo JSON objects.

GEOCODE	DESCRIPTION	OUTPUT COLUMN NAME
STA	State code	STA_GCODE
STAN	State Name	STA_GEOGRAPHIC_NAME
CTY	County code	CTY_GCODE
CTYN	County name	CTY_GEOGRAPHIC_NAME
TRA	Tract code	TRA_GCODE
BGR	Block group code	BGR_GCODE
CSA	Combined Statistical Area Code	CSA_GCODE
CSAN	Combined Statistical Area Name	CSA_GEOGRAPHIC_NAME
CBS	Core Based Statistical Area Code	CBS_GCODE
CBSN	Core Based Statistical Area Name	CBS_GEOGRAPHIC_NAME

GEOCODE	DESCRIPTION	OUTPUT COLUMN NAME
DMA	Nielsen Designated Market Area Code	DMA_GCODE
DMAN	Nielsen Designated Market Area Name	DMA_GEOGRAPHIC_NAME
PLA	Place Code	PLA_GCODE
PLAN	Place Name	PLA_GEOGRAPHIC_NAME
MCD	Minor Civil Division Code	MCD_GCODE
MCDN	Minor Civil Division Name	MCD_GEOGRAPHIC_NAME
TDZ	Three Digit ZIP Code	TDZ_GCODE
CNG	Congressional District Code	CNG_GCODE
CNGN	Congressional District Name	CNG_GEOGRAPHIC_NAME

Sample Java Code for Calling Report Services

Below are the following: a java file and a pom.xml file that combines all the Java® code snippets referenced throughout the document. You can generate the structure of the project using Spring Boot's Spring Initializer and make use of the linked files to test the examples.

DemoApplication

pom

TECHNICAL SUPPORT

If you need further assistance, not provided in the guide, please contact the Claritas Solution Center between 9:00 a.m. and 8:00 p.m. (Monday through Friday, EST) at 800.866.6511

LEGAL NOTIFICATIONS

Business-Facts, ConneXions, Pop-Facts, PRIZM and P$YCLE are registered trademarks of Claritas, LLC. Consumer Buying Power and Retail Market Power are registered trademarks of Environics Analytics. The DMA data are proprietary to The Nielsen Company (US), LLC ("Nielsen"), a Third-Party Licensor, and consist of the boundaries of Nielsen's DMA regions within the United States of America. Other company names and product names are trademarks or registered trademarks of their respective companies and are hereby acknowledged.

This documentation contains proprietary information of Claritas. Publication, disclosure, copying, or distribution of this document or any of its contents is prohibited, unless consent has been obtained from Claritas.

Some of the data in this document is for illustrative purposes only and may not contain or reflect the actual data and/or information provided by Claritas to its clients.