Sites USA API — Quick Start Guide

A plain-English walkthrough of the 13 most commonly used calls in the Sites USA / REGIS Online public API — for Account Executives on sales calls and for prospective clients' technical contacts.

BASE URL `https://api.sitesusa.com/`

All requests and responses are JSON unless a section says otherwise.

How to use this guide

You don't need to be a developer to follow this. Each section below answers one question — "how do I get X?" — and gives you:

- **What it's for**, in plain language
- The **endpoint** (method + path)
- The **request** you send (with a real example)
- The **response** you get back
- A **cURL** example (copy/paste into any terminal) and a **JavaScript** example (copy/paste into any Node.js or browser script)

If a prospect's developer asks "does your API do X?" — find the matching number below and you have a working example in under a minute.

Quick glossary

Term	Plain-English meaning
JWT	Your temporary login token. You trade your API key for one of these once, then reuse it.
WKT	"Well-Known Text" — a plain-text way of describing a shape (a circle, a polygon) on a map. Most of our endpoints take a trade area as a WKT shape.
Trade Area	The geographic area you're analyzing — a radius ring, a drive-time area, or a standard boundary like a county or MSA.
KTA	"Known Trade Area" — one of our pre-built standard boundaries (Zip, City, County, MSA, State, Nation) instead of a custom radius or drive time.
Void Report	A report that compares a large area to a small area and flags retailers present in the large area but *missing* ("void") from the small one — a classic gap-analysis tool for site selectors.

Before you start: Authentication

Every call below (except login itself) requires a **Bearer token** in the request header:

```
Authorization: Bearer <your JWT>
```

You get that JWT from the login call in Section 1. **It's valid for 12 hours** — get it once, reuse it for every call that day. Don't call login before every request; you'll hit the rate limit and it's unnecessary.

Your **API Access Token** (the credential you exchange *for* the JWT) is provided by your Sites USA account contact during onboarding — it's different from the JWT and doesn't expire hourly.

Common Workflows

A typical site-evaluation workflow touches several of these calls in sequence:

1. **Login** (#1) → get your JWT, reuse for the day
2. **Build Trade Areas** (#2) → get WKT shapes for a radius, drive time, and/or census boundary around the site
3. Feed those shapes into any combination of:
   - **Demographics** (#5) using variable IDs from #3
   - **Merchant Counts** (#10) using category/merchant IDs from #4

- **Demographic Report** (#7) or **Void Report** (#8) for a polished, shareable PDF

4. Layer in **Traffic Counts** (#9) and **Mobile Visits** (#13) for the specific point
5. Use **Expansion Rates** (#11) and **Cotenants** (#12) for market-level or brand-level context

Sections 3, 4, and 6 (variables, merchant categories, report templates) are typically called **once** and cached — they rarely change and don't need to be re-fetched per site.

---

01 Login — Get Your Token

What it's for: The first call in every session. Trade your API Access Token for a JWT that authenticates everything else you do for the next 12 hours.

ENDPOINT	`POST /api/v1/login`
RATE LIMIT	1 request / second

REQUEST BODY

```
{
  "value": "<your API Access Token>"
}
```

Response: A JWT string (valid 12 hours).

CURL

```
curl -X POST "https://api.sitesusa.com/api/v1/login" \
  -H "Content-Type: application/json" \
  -d '{"value": "YOUR_API_ACCESS_TOKEN"}'
```

JAVASCRIPT

```
async function login(apiAccessToken) {
  const res = await fetch("https://api.sitesusa.com/api/v1/login", {
    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify({ value: apiAccessToken })
  });
  const jwt = await res.json();
  return jwt; // cache this — reuse for up to 12 hours
}
```

TIP FOR THE CALL

This is the easiest thing to demo live — "watch, one call, and we're authenticated for the rest of the day."

02 Building Trade Areas — Radius, Drive Time, or Census Boundary

What it's for: Before you can pull demographics, run a report, or count merchants, you need to define the *area* you're analyzing. This one endpoint builds all three types of trade areas and hands back the map shape (WKT) you'll plug into later calls.

ENDPOINT `POST /api/v1/TradeAreas`

RATE LIMIT 1 request / 5 seconds

There are three ways to define an area in the same request:

Type	How to specify it
Radius ring	`polygonType: 1`, `size` = miles (e.g., `1` for a 1-mile ring)
Drive time	`polygonType: 2`, `size` = minutes (e.g., `5` for a 5-minute drive time). Optional `driveTimeSettings` control traffic direction, day/time, tolls, and ferries.
Census boundary (KTA)	`knownType` + `knownTypeId` identify the specific boundary (a county, MSA, state, etc. — see below for how to find these IDs)

REQUEST BODY — ONE RADIUS RING, ONE DRIVE TIME, ONE MSA BOUNDARY

```
{
  "coordinate": { "x": -111.9119726, "y": 33.3047802 },
  "driveTimeSettings": {
    "avoidFerries": false,
    "avoidTolls": false,
    "isOutbound": true,
    "isDtCustom": false,
    "dayOfWeek": -1,
    "timeOfDay": -1,
    "costFactor": 1
  },
  "tradeAreas": [
    {
      "polygonType": 1,
      "size": 1,
      "knownType": -1,
      "knownTypeId": -1,
      "coordinate": { "x": -111.9119726, "y": 33.3047802 },
      "tradeAreaDisplayName": "1 mi Radius Ring"
    },
    {
      "polygonType": 2,
      "size": 5,
      "knownType": -1,
      "knownTypeId": -1,
      "coordinate": { "x": -111.9119726, "y": 33.3047802 },
      "tradeAreaDisplayName": "5 min Drive Time"
    },
    {
      "polygonType": -1,
      "size": 0,
      "knownType": 2,
      "knownTypeId": 679,
      "coordinate": { "x": -111.9119726, "y": 33.3047802 },
      "tradeAreaDisplayName": "Phoenix-Mesa-Chandler MSA"
    }
  ]
}
```

Response: An array of trade areas, each now filled in with a `shape` (WKT) — this is what you feed into Demographics, Reports, Void Reports, and Merchant Counts.

CURL

```
curl -X POST "https://api.sitesusa.com/api/v1/TradeAreas" \
  -H "Authorization: Bearer $JWT" \
  -H "Content-Type: application/json" \
  -d @tradeareas_request.json
```

JAVASCRIPT

```
async function buildTradeAreas(jwt, requestBody) {
  const res = await fetch("https://api.sitesusa.com/api/v1/TradeAreas", {
    method: "POST",
    headers: {
      "Authorization": `Bearer ${jwt}`,
      "Content-Type": "application/json"
    },
    body: JSON.stringify(requestBody)
  });
  return res.json(); // TradeArea[] — each with a "shape" (WKT) field
}
```

Finding the right census boundary ID

Don't guess the `knownType`/`knownTypeId` — look them up:

- `GET /api/v1/TradeAreas/kta` — lists every KTA type available to your account (Zip, City, County, MSA, State, Nation, etc.) with its numeric type ID.
- `GET /api/v1/TradeAreas/kta/{lat}_{lon}` — for a specific point, returns the actual county, city, zip, MSA, state, and nation names/IDs that contain it (e.g., "this point is in Maricopa County, the Phoenix-Mesa-Chandler MSA, and Arizona"). This is the fastest way to auto-populate the correct `knownTypeId` for a site.

CURL — POINT LOOKUP

```
curl "https://api.sitesusa.com/api/v1/TradeAreas/kta/33.3186_-111.9041" \
  -H "Authorization: Bearer $JWT"
```

03 Get All Available Demographic Variables

What it's for: A one-time call to see every demographic variable your account can request (population, income, age brackets, daytime employment, etc.) along with the numeric ID you'll use to request it later. Most integrations call this once and cache the list.

ENDPOINT	`GET /api/v1/Demographics/variables`
RATE LIMIT	1 request / 5 seconds

RESPONSE

```
[
  { "id": 121,   "name": "Median Household Income" },
  { "id": 16286, "name": "Total Population" }
]
```

CURL

```
curl "https://api.sitesusa.com/api/v1/Demographics/variables" \
  -H "Authorization: Bearer $JWT"
```

JAVASCRIPT

```
async function getVariables(jwt) {
  const res = await fetch("https://api.sitesusa.com/api/v1/Demographics/variables", {
    headers: { "Authorization": `Bearer ${jwt}` }
  });
  return res.json(); // [{ id, name }, ...]
}
```

04 Get All Merchant Categories and Merchant IDs

What it's for: Another one-time reference call. Returns every merchant category (e.g., "Restaurants Fast Food Major") with its category ID, and every individual merchant/brand inside that category with its merchant ID. You'll reuse these IDs constantly — for Void Reports, merchant counts, expansion rates, and cotenant lookups.

ENDPOINT	`GET /api/v1/MerchantCategories`
RATE LIMIT	1 request / second

RESPONSE SHAPE

```
[
  {
    "id": "1360",
    "name": "Restaurants Fast Food Major",
    "merchants": [
      { "id": "2445", "name": "Taco Bell" },
      { "id": "2441", "name": "McDonald's" }
    ]
  }
]
```

IDs above are illustrative — pull the real ones from this call.

CURL

```
curl "https://api.sitesusa.com/api/v1/MerchantCategories" \
  -H "Authorization: Bearer $JWT"
```

JAVASCRIPT

```
async function getMerchantCategories(jwt) {
  const res = await fetch("https://api.sitesusa.com/api/v1/MerchantCategories", {
    headers: { "Authorization": `Bearer ${jwt}` }
  });
  return res.json(); // [{ id, name, merchants: [{ id, name, ... }] }]
}
```

05 Process Demographics for a Trade Area

What it's for: The core demographic pull. Takes the WKT shape(s) you got from Section 2 and the variable IDs you got from Section 3, and returns the raw values for each.

ENDPOINT	`POST /api/v1/Demographics`
RATE LIMIT	1 request / 5 seconds

REQUEST BODY

```
{
  "variableIds": [16286, 121],
  "wkts": [
    { "id": 1, "name": "1 mile radius ring", "shape": "POLYGON((...))" },
    { "id": 2, "name": "3 mile radius ring", "shape": "POLYGON((...))" }
  ]
}
```

The shape values here are exactly the shape/WKT strings returned from the Trade Areas call in Section 2.

RESPONSE — ONE OBJECT PER WKT SUBMITTED

```
[
  {
    "id": 1,
    "name": "1 mile radius ring",
    "demographics": [
      { "id": 16286, "value": 12345.0 },
      { "id": 121,   "value": 72000.0 }
    ]
  }
]
```

NOTE

The response gives you variable **IDs** and values, not names — keep a local map of ID → name from Section 3's response.

CURL

```
curl -X POST "https://api.sitesusa.com/api/v1/Demographics" \
  -H "Authorization: Bearer $JWT" \
  -H "Content-Type: application/json" \
  -d @demographics_request.json
```

JAVASCRIPT

```
async function getDemographics(jwt, variableIds, wkts) {
  const res = await fetch("https://api.sitesusa.com/api/v1/Demographics", {
    method: "POST",
    headers: {
      "Authorization": `Bearer ${jwt}`,
      "Content-Type": "application/json"
    },
    body: JSON.stringify({ variableIds, wkts })
  });
  return res.json();
}
```

06 Get All Available Demographic Report Templates

What it's for: A one-time call to see which pre-built, formatted PDF report templates your account has access to. `RFULL5` (our common 5-page demographic report) is one example — this call shows every template you're licensed for.

ENDPOINT	`GET /api/v1/Reports/demographic/templates`
RATE LIMIT	1 request / 30 seconds

RESPONSE

```
[
  { "id": 4, "name": "5-Page Full Demographic Report", "templateName": "RFULL5",
"hasAccess": true, "datasetId": 1 }
]
```

CURL

```
curl "https://api.sitesusa.com/api/v1/Reports/demographic/templates" \
  -H "Authorization: Bearer $JWT"
```

JAVASCRIPT

```
async function getReportTemplates(jwt) {
  const res = await fetch("https://api.sitesusa.com/api/v1/Reports/demographic/templates",
{
    headers: { "Authorization": `Bearer ${jwt}` }
  });
  return res.json(); // [{ id, name, templateName, hasAccess, datasetId }]
}
```

07 Run a Demographic Report

What it's for: Generates the actual formatted report file (PDF by default) using a template from Section 6, branding/title info, and the trade area(s) to report on.

ENDPOINT	`POST /api/v1/Reports/demographic/{templateName}`
QUERY PARAM	`format` (default `pdf`)
RATE LIMIT	1 request / 30 seconds

REQUEST BODY

```
{
  "title": "Chandler Blvd Site",
  "subTitle": "5-Mile Trade Area Overview",
  "preparedFor": "Prospect Name",
  "preparedBy": "Your Name",
  "latitude": 33.3048,
  "longitude": -111.9145,
  "logo": null,
  "tradeAreas": [
    {
      "id": 1,
      "name": "1 mi Radius",
      "wkt": "POLYGON((...))",
      "typeId": 1,
      "size": 1,
      "ktaTypeId": -1,
      "ktaId": -1
    }
  ]
}
```

`wkt` here comes straight from the Trade Areas call in Section 2.

Response: The completed report file as binary data (PDF by default).

CURL

```
curl -X POST "https://api.sitesusa.com/api/v1/Reports/demographic/RFULL5?format=pdf" \
  -H "Authorization: Bearer $JWT" \
  -H "Content-Type: application/json" \
  -d @report_request.json \
  --output report.pdf
```

JAVASCRIPT

```
async function runDemographicReport(jwt, templateName, requestBody) {
  const res = await fetch(
    `https://api.sitesusa.com/api/v1/Reports/demographic/${templateName}?format=pdf`,
    {
      method: "POST",
      headers: {
        "Authorization": `Bearer ${jwt}`,
        "Content-Type": "application/json"
      },
      body: JSON.stringify(requestBody)
    }
  );
  const blob = await res.blob(); // save or stream this as a .pdf
  return blob;
}
```

08 Run a Void Report

What it's for: A gap-analysis report. You give it a **large "market" trade area** (e.g., an MSA or county) and a **small "comparison" trade area** (e.g., a 3-mile ring around a site), plus a set of merchants and/or categories. The report flags any retailer that's present in the market area but has **zero locations** in the comparison area — a "void," and a classic pitch for "here's what's missing near this site." `ADVANCED_DETAIL_VOID` is a common template name.

ENDPOINT	`POST /api/v1/VoidReports/{templateName}`
QUERY PARAM	`format` (default `pdf`)
TEMPLATES	`GET /api/v1/VoidReports/templates`
RATE LIMIT	1 request / 30 seconds

REQUEST BODY

```
{
  "title": "Void Analysis — Gilbert Site",
  "subtitle": "Fast Food Gap Analysis",
  "preparedFor": "Prospect Name",
  "preparedBy": "Your Name",
  "latitude": 33.3048,
  "longitude": -111.9145,
  "logo": null,
  "merchantIds": [2445],
  "categoryIds": [1360],
  "marketTradeArea": {
    "id": 1,
    "name": "Maricopa County, AZ",
    "wkt": "POLYGON((...))",
    "typeId": 1,
    "size": 1,
    "ktaTypeId": -1,
    "ktaId": -1
  },
  "comparisonTradeArea": {
    "id": 2,
    "name": "Gilbert, AZ 3-mile ring",
    "wkt": "POLYGON((...))",
    "typeId": 1,
    "size": 1,
    "ktaTypeId": -1,
    "ktaId": -1
  }
}
```

CURL

```
curl -X POST "https://api.sitesusa.com/api/v1/VoidReports/ADVANCED_DETAIL_VOID?format=pdf"
\
  -H "Authorization: Bearer $JWT" \
  -H "Content-Type: application/json" \
  -d @void_request.json \
  --output void_report.pdf
```

JAVASCRIPT

```
async function runVoidReport(jwt, templateName, requestBody) {
  const res = await fetch(
    `https://api.sitesusa.com/api/v1/VoidReports/${templateName}?format=pdf`,
    {
      method: "POST",
      headers: {
        "Authorization": `Bearer ${jwt}`,
        "Content-Type": "application/json"
      },
      body: JSON.stringify(requestBody)
    }
  );
  return res.blob();
}
```

09 Traffic Counts for the Nearest Intersection

What it's for: Given a coordinate, returns the closest traffic-counted intersection and its counts on both cross streets — useful for quickly qualifying visibility/access for a site.

ENDPOINT	`GET /api/v1/TrafficCounts/intersections/{lat},{lon}`
QUERY PARAM	`buffer` — search radius in miles, between 0.1 and 0.5 (default 0.25)
RATE LIMIT	1 request / second

RESPONSE

```
{
  "intersectionOneName": "Chandler Blvd",
  "intersectionTwoName": "Alma School Rd",
  "intersectionOneCount": 28500,
  "intersectionTwoCount": 31200,
  "intersectionLatitude": 33.3049,
  "intersectionLongitude": -111.9143,
  "distanceFromPoint": "0.08 mi"
}
```

CURL

```
curl "https://api.sitesusa.com/api/v1/TrafficCounts/intersections/33.3048,-111.9145?
buffer=0.25" \
  -H "Authorization: Bearer $JWT"
```

JAVASCRIPT

```
async function getTrafficCounts(jwt, lat, lon, buffer = 0.25) {
  const res = await fetch(
    `https://api.sitesusa.com/api/v1/TrafficCounts/intersections/${lat},${lon}?
buffer=${buffer}`,
    { headers: { "Authorization": `Bearer ${jwt}` } }
  );
  return res.json();
}
```

NOTE

A `400` response here generally means no counted intersection was found within the buffer — treat it as "no data," not a failure, and consider retrying with a larger buffer (up to 0.5 mi).

10 Merchant Counts in a Trade Area

What it's for: "How many of X are in this area?" — pass a trade area shape plus a list of merchant IDs and/or category IDs, get counts back. Works with radius, drive time, or census boundary shapes from Section 2.

ENDPOINT	`POST /api/v1/Merchants/counts`
QUERY PARAMS	`merchantIds` and/or `categoryIds` (comma-separated)
BODY	An array of WKT strings — plain strings here, not the `{id, name, shape}` objects used in Demographics
RATE LIMIT	1 request / second

REQUEST

```
POST /api/v1/Merchants/counts?categoryIds=1360,1368

["POLYGON((-111.90 33.25,-111.70 33.25,-111.70 33.40,-111.90 33.40,-111.90 33.25))"]
```

RESPONSE — ONE ARRAY OF COUNTS PER WKT SUBMITTED

```
[
  [
    { "id": 1360, "name": "Restaurants Fast Food Major", "value": 14 },
    { "id": 1368, "name": "Restaurants Fast Food Minor", "value": 9 }
  ]
]
```

CURL

```
curl -X POST "https://api.sitesusa.com/api/v1/Merchants/counts?categoryIds=1360,1368" \
  -H "Authorization: Bearer $JWT" \
  -H "Content-Type: application/json" \
  -d '["POLYGON((-111.90 33.25,-111.70 33.25,-111.70 33.40,-111.90 33.40,-111.90
33.25))"]'
```

JAVASCRIPT

```
async function getMerchantCounts(jwt, wktArray, { merchantIds, categoryIds } = {}) {
  const params = new URLSearchParams();
  if (merchantIds) params.set("merchantIds", merchantIds.join(","));
  if (categoryIds) params.set("categoryIds", categoryIds.join(","));

  const res = await fetch(
    `https://api.sitesusa.com/api/v1/Merchants/counts?${params}`,
    {
      method: "POST",
      headers: {
        "Authorization": `Bearer ${jwt}`,
        "Content-Type": "application/json"
      },
      body: JSON.stringify(wktArray)
    }
  );
  return res.json();
}
```

11 Merchant Expansion Rates

What it's for: Shows whether a retailer (or category) is growing or shrinking its footprint within a large standard boundary — MSA, State, or Nation only (not available for custom radius/drive-time areas).

ENDPOINT	`GET /api/v1/Merchants/expansion/{ktaType}`
PATH PARAM	`ktaType` — boundary type (confirmed: `2` = MSA, `7` = Nation; call `GET /api/v1/TradeAreas/kta` for the full list, including State)
QUERY PARAMS	`ktaId` (required for MSA/State — the specific area's ID, e.g., from the KTA lookup in Section 2; omit for Nation), `merchantIds`, `categoryIds`
RATE LIMIT	1 request / second

RESPONSE

```
[
  [
    {
      "ktaId": 38,
      "ktaType": 2,
      "name": "Phoenix-Mesa-Chandler MSA",
      "previousCount": 42,
      "currentCount": 51,
      "expansionRate": 0.214
    }
  ]
]
```

CURL — MSA EXAMPLE

```
curl "https://api.sitesusa.com/api/v1/Merchants/expansion/2?
ktaId=38&categoryIds=1360,1368&merchantIds=2441,2445" \
  -H "Authorization: Bearer $JWT"
```

CURL — NATION-WIDE EXAMPLE

```
curl "https://api.sitesusa.com/api/v1/Merchants/expansion/7?
categoryIds=1360,1368&merchantIds=2441,2445" \
  -H "Authorization: Bearer $JWT"
```

JAVASCRIPT

```
async function getExpansionRates(jwt, ktaType, { ktaId, merchantIds, categoryIds } = {}) {
  const params = new URLSearchParams();
  if (ktaId) params.set("ktaId", ktaId);
  if (merchantIds) params.set("merchantIds", merchantIds.join(","));
  if (categoryIds) params.set("categoryIds", categoryIds.join(","));

  const res = await fetch(
    `https://api.sitesusa.com/api/v1/Merchants/expansion/${ktaType}?${params}`,
    { headers: { "Authorization": `Bearer ${jwt}` } }
  );
  return res.json();
}
```

12 Top Merchant Cotenants

What it's for: "Who does this retailer usually locate next to?" Pass a merchant and/or category, get back a ranked list of the other merchants most frequently found co-located with it — useful for pitching a site to a prospect by showing who their likely neighbors would be.

ENDPOINT	`GET /api/v1/Merchants/cotenants`
QUERY PARAMS	`merchantIds` and/or `categoryIds` (comma-separated)
RATE LIMIT	1 request / second

RESPONSE

```
[
  [
    {
      "merchantId": 2445,
      "merchantName": "Taco Bell",
      "cotenants": [
        { "id": "2441", "value": 0.62 },
        { "id": "1180", "value": 0.48 }
      ]
    }
  ]
]
```

`cotenants[].id` maps to a merchant ID; `value` reflects relative co-location strength.

CURL

```
curl "https://api.sitesusa.com/api/v1/Merchants/cotenants?
categoryIds=1360,1368&merchantIds=2445" \
  -H "Authorization: Bearer $JWT"
```

JAVASCRIPT

```
async function getCotenants(jwt, { merchantIds, categoryIds } = {}) {
  const params = new URLSearchParams();
  if (merchantIds) params.set("merchantIds", merchantIds.join(","));
  if (categoryIds) params.set("categoryIds", categoryIds.join(","));

  const res = await fetch(
    `https://api.sitesusa.com/api/v1/Merchants/cotenants?${params}`,
    { headers: { "Authorization": `Bearer ${jwt}` } }
  );
  return res.json();
}
```

13 Mobile Visits for a Merchant Location

What it's for: Foot-traffic data for a specific store location — monthly visit counts, visitor counts, median daily visits, and similar nearby brands, over the last 12 months.

ENDPOINT	`GET /api/v1/MobileData/{lat},{lon}/merchant/{merchantId}`
COORDINATE	Must be within 0.1 miles of the actual store
MERCHANTID	A Sites USA-tracked merchant ID (from Section 4)
RATE LIMIT	1 request / second

RESPONSE — GEOJSON


```
{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "geometry": { "type": "Point", "coordinate": { "x": -111.9119726, "y": 33.3047802 }
},
      "properties": {
        "name": "Albertsons",
        "address": "4060 W Ray Rd",
        "city": "Chandler",
        "state": "AZ",
        "zip": "85226",
        "startdate": "6-1-2022",
        "totalvisits": [26909, "...", 28001],
        "totalvisitors": [26909, "...", 25001],
        "mediandailyvisits": [893, "...", 930],
        "brands": ["Fry's Food and Drug Stores"]
      }
    }
  ]
}
```


CURL

```
curl "https://api.sitesusa.com/api/v1/MobileData/33.3928,-112.0977/merchant/2445" \
  -H "Authorization: Bearer $JWT"
```

JAVASCRIPT

```
async function getMobileVisits(jwt, lat, lon, merchantId) {
  const res = await fetch(
    `https://api.sitesusa.com/api/v1/MobileData/${lat},${lon}/merchant/${merchantId}`,
    { headers: { "Authorization": `Bearer ${jwt}` } }
  );
  return res.json();
}
```

NOTE

A `400` response here means no mobile data was found for that merchant/location pairing — most often because the coordinate isn't within 0.1 miles of the actual store.

Appendix A: Rate Limits at a Glance

Speed tier	Limit	Applies to
Default	1 request / second	Login, Merchant Categories, Merchant Counts, Merchant Expansion, Cotenants, Traffic Counts, Mobile Data
Extended	1 request / 5 seconds	Trade Areas (build + KTA lookups), Demographics (variables list + processing)
Max	1 request / 30 seconds	Report templates, running a Demographic Report, Void Report templates, running a Void Report

A `429` response means you've hit the limit — the response includes a `Retry-After` header telling you how many seconds to wait.