Data247 API Version 3.0

User's Manual

Data247 has a single REST-based API which works for all of our data services. It's easy to use and yet very powerful, allowing you to access multiple Data247 services in a single API call.

TABLE OF CONTENTS

API REQUEST FORMAT

Here are some sample API requests (Available via HTTPS GET or POST):

- https://api.data247.com/v3.0?key=YOURAPIKEY&api=MT&phone=6171234567&addfields=last_port_date
- https://api.data247.com/v3.0?key=YOURAPIKEY&api=AP&fname=Joe&lname=Smith&address=24 Main Street&City=Boston&state=MA&zip=02134&out=yaml
- https://api.data247.com/v3.0?key=YOURAPIKEY&api=CU,AN,APR&phone=16172223333&out=xml

There are 6 main parts to the API queries:

- The URL: This remains the same for each query: https://api.data247.com/v3.0?
- The API key: Each API user has their own unique API key(s) which identifies them. If you don't know your API key, log into your Data247 account, go to the "Account" menu, and select "API Key / Password". If you don't see an API key in the list on this page, you can generate a new one by entering a description and then pressing the "Generate API Key" button.
- The API Service Code(s): This identifies the particular API data service or services you wish to use. Data247 offers many different data services such as Text@, phone append, IP Geolocation, etc. For the full list, see our "Data247 Data Services" section below.
- The Input Data: For each Data247 service, you provide input data and we reply with a response based on your input. The input parameters will vary for each service. See the "Data247 Data Services" section below for more information.
- The Field Specifiers: There are two types of field specifiers; "fields=" and "addfields=". The former specifies ALL data fields to be returned by the query, and the latter specifies which optional fields should be returned alongside the standard return fields. In both cases, the fields should be separated by commas. Field specifiers are optional. If omitted, only the standard data fields will be returned.
- The Response Format: Either xml, json, yaml, toml, msgpack, or csv. See the "API RESPONSE FORMATS" section for more details. The response format specifier is optional. If omitted, JSON format will be returned by default.

Note: To do multiple queries in a single API call, see the "MULTIPLE QUERIES IN A SINGLE API CALL" section below.

API RESPONSE FORMATS

The API can currently support 6 different output formats: JSON, XML, YAML, TOML, MsgPack, and CSV. JSON is the default response format.

You can specify the API response format using the “out=” parameter in the API request, as follows (note they are all lower-case):

- out=json
- out=xml
- out=yaml
- out=toml
- out=csv
- out=msgpack

JSON RESPONSES

A JSON response encapsulates the entire response in a “response” tag. Inside the response tag will always be a “status” indicator. If the status is anything other than “OK”, the query failed. See the “API ERROR MESSAGES” section below for more information on errors. If the status is “OK”, then the results from the query will appear as an array of “results”, as shown below:


```
{
 "response": {
   "status": "OK",
   "results": [
    {
      "phone": "15085551212",
      "wless": "n",
      "carrier_name": "Neutral Tandem-Illinois, LLC - IL",
      "carrier_id": 50724,
      "sms_address": "",
      "mms_address": ""
    }
  ]
 }
}
```


XML RESPONSES

An XML response encapsulates the entire response in a “<response>” tag. Inside the response tag will always be a “status” indicator. If the status is anything other than “OK”, the query failed. See the “API ERROR MESSAGES” section below for more information on errors. If the status is “OK”, then the results from the query will appear as an array of “<results>”, each encapsulated in a “<result>” tag, as shown below:

```
<response>
  <status>OK</status>
  <results>
    <result>
      <phone>1505551212</phone>
      <wless>n</wless>
      <carrier_name>Neutral Tandem-Illinois, LLC - IL</carrier_name>
      <carrier_id>50724</carrier_id>
      <sms_address/>
      <mms_address/>
    </result>
  </results>
</response>
```

CSV RESPONSES

A CSV response is similar to a .csv (comma separated values) file format. It contains a header row with each column name, in quotes, separated by commas. Similarly, each result will appear in its own row, with each data element in quotes, separated by commas.

```
"phone","wless","carrier_name","carrier_id","sms_address","mms_address"
"15084041018","n","Neutral Tandem-Illinois, LLC - IL","50724","",""
```

YAML RESPONSES

A YAML response contains the entire response under the “response” tag. Inside the response tag will always be a “status” element. If the status is anything other than “OK”, the query failed. See the “API ERROR MESSAGES” section below for more information on errors. If the status is “OK”, then the results from the query will appear as an array of ‘results’, as shown below:

```
response:
  results:
  - carrier_id: 40654
    carrier_name: Sprint Spectrum, L.P.
    mms_address: 7818641234@tmomail.net
    phone: '17818641234'
    sms_address: 7818641234@tmomail.net
    wless: y
  status: OK
```

TOML RESPONSES

A TOML response contains the entire response under the “response” tag. Inside the response tag will always be a “status” element. If the status is anything other than “OK”, the query failed.

See the “API ERROR MESSAGES” section below for more information on errors. If the status is “OK”, then the results from the query will appear as an array of ‘results’, as shown below:

```
[response]
status = "OK"
[[response.results]]
phone = "17818641234"
wless = "y"
carrier_name = "Sprint Spectrum, L.P."
carrier_id = 40654
sms_address = "7818641234@tmomail.net"
mms_address = "7818641234@tmomail.net"
```

MSGPACK RESPONSES

MsgPack (https://msgpack.org/) uses binary characters as field separators, and tends to be more compact than other formats. The MsgPack response contains the entire response under the “response” tag. Inside the response tag will always be a “status” element. If the status is anything other than “OK”, the query failed. See the “API ERROR MESSAGES” section below for more information on errors. If the status is “OK”, then the results from the query will appear as an array of ‘results’, as shown below:

```
\x81\xa8response\x82\xa6status\xa2OK\xa7results\x91\x86\xa5phone\xab17818641234\xa5wless\xa1y\xaccarrier_name\xbeBellsouth Mobility, LLC
(AT+T)\xaacarrier_id\x06\xabsms_address\xb67818641234@txt.att.net\xabmms_address\xb67818641234@mms.att.net
```

DATA247 DATA SERVICES

The various Data247 services are listed below, alphabetically:

Append247 (Email)

Description: Returns the email address associated with a name and address (business or residential)
Service Code: AE
Input Data Fields:

- fname: Person's first name
- lname: Person's last name
- name: Full name; may be used instead of fname and name
- business_name - If business address (optional)
- address: Contact's street address
- city: Contact's city
- state: Contact's state
- zip: Contact's zip code (optional)
- phone: Contact's phone number (optional)

Standard Return Fields:

- firstname: Contact's first name
- lastname: Contact's last name
- address: Contact's address
- city: Contact's city
- state: Contact's state
- zip: Contact's zip code
- email: Contact's email address

Optional Return Fields:

- billable - Indicates whether customer was charged for this query
- cost - Cost of this query
- website: Company website (business listings only)
- num_employees: Total employees (business listings only)
- total_revenue: Total revenue (business listings only)
- title: Contact's business title (business listings only)
- business_type: Type of business (business listings only)

Append247 (Gender)

Description: Returns a person's probable gender based on their first (given) name
Service Code: AG
Input Data Fields:

- fname - Person's first name

- name - Person's full name

Standard Return Fields:

- fname - Person's first name
- gender - Probable gender (MALE or FEMALE)
- gender_pct - The percent likelihood of gender being correct.

Optional Return Fields:

- billable - Indicates whether customer was charged for this query
- cost - Cost of this query

Append247 (Name)

Description: Returns the name associated with a phone number
Service Code: AN
Input Data Fields:

- phone: phone number

Standard Return Fields:

- name - Person's first name

Optional Return Fields:

- billable - Indicates whether customer was charged for this query
- cost - Cost of this query

Append247 (Phone)

Description: Returns the phone number(s) associated with a name and address (business or residential)
Service Code: AP
Input Data Fields:

- fname: Person's first name
- lname: Person's last name
- name: Full name; may be used instead of fname and name
- business_name - If business address (optional)
- address: Contact's street address
- city: Contact's city
- state: Contact's state
- zip: Contact's zip code (optional)
- email: Contact's email address (optional)

Standard Return Fields:

- firstname: Contact's first name
- lastname: Contact's last name
- address: Contact's address
- city: Contact's city
- state: Contact's state

- zip: Contact's zip code
- phone: Contact's phone number
- type: Type of phone (Landline, Wireless, or VOIP)
- addtl: An array of up to five additional phone numbers with phone type.

Optional Return Fields:

- billable - Indicates whether customer was charged for this query
- cost - Cost of this query
- website: Company website (business listings only)
- num_employees: Total employees (business listings only)
- total_revenue: Total revenue (business listings only)
- title: Contact's business title (business listings only)
- business_type: Type of business (business listings only)

Append247 (Property Data)

Description: Returns a property's home stats, ownership information, financials and foreclosure data

Service Code: ARE

Input Data Fields:

- address: Property street address (USA only)
- city: Property city (USA only)
- state: Property state (USA only)
- zip: Property zip code (optional, USA only)

Standard Return Fields:

- owner_name: Property owner's name
- property_address: Property owner's street address
- property_city: Property owner's city
- property_state: Property owner's state
- property_zip: Property owner's zip code
- mailing_address: Property owner's mailing address
- mailing_city: Property owner's mailing city
- mailing_state: Property owner's mailing state
- mailing_zip: Property owner's mailing zip code
- beds: Number of bedrooms
- baths: Number of bathrooms
- building_area: Building area (sq ft)
- lot_area: Lot area (sq ft)
- assessed_value: Assessed value of property
- year_built: Year the property was built
- latest_sale_price: Last sale price
- latest_sale_recording_date: Last sale recording date
- is_property_in_foreclosure: (Y/N)
- foreclosure_recording_date: Foreclosure date (returns null if property isn't in foreclosure)

- apn: Assessor's Parcel Number
- land_type: (e.g., Single Family Residential, Commercial)
- equity: Equity percentage

Optional Return Fields:

- billable - Indicates whether customer was charged for this query
- cost - Cost of this query

Append247 (Reverse Email)

Description: Returns the name and address associated with an email address.
Service Code: AER
Input Data Fields:

- email: Contact’s email address

Standard Return Fields:

- phone - Contact’s phone number
- email - Contact’s email address
- firstname - Contact’s first name
- lastname - Contact’s last name
- m_initial - Contact’s middle initial
- business - Contact’s business name
- address - Contact’s address
- city - Contact’s city
- state - Contact’s state
- zip - Contact’s zip code
- zip4 - Additional 4 digits of contact’s zip code
- addrtype - business or residential
- gender - MALE or FEMALE
- gender_prcnt - Percent likelihood of gender being correct

Optional Return Fields:

- billable - Indicates whether customer was charged for this query
- cost - Cost of this query

Append247 (Reverse Phone)

Description: Returns the name and address associated with a phone number.
Service Code: APR
Input Data Fields:

- phone: Contact’s phone number

Standard Return Fields:

- phone - Contact’s phone number
- email - Contact’s email address
- firstname - Contact’s first name
- lastname - Contact’s last name

- m_initial - Contact's middle initial
- business - Contact's business name
- address - Contact's address
- city - Contact's city
- state - Contact's state
- zip - Contact's zip code
- zip4 - Additional 4 digits of contact's zip code
- addrtype - business or residential
- gender - MALE or FEMALE
- gender_prcnt - Percent likelihood of gender being correct

Optional Return Fields:

- billable - Indicates whether customer was charged for this query
- cost - Cost of this query

Append247 (Reverse Zipcode)

Description: Returns various information about a zipcode
Service Code: AZR
Input Data Fields:

- zip - Either a 5 or 9 digit zipcode (NNNNN or NNNNN-NNNN)

Standard Return Fields:

- zip - Zip code
- zip4 - Last 4 digits of zipcode
- streetnum - Street number or number range of address
- predir - Direction indicator before street name
- street - The street name of the address
- street_suffix - Like St, Rd, Pl, etc.
- postdir - Direction indicator after street name
- unit - Unit number or range of street address
- city - Name of city
- state - Name of state
- county - Name of county
- PRUrbCode" - Puerto Rico Urbanization Code
- country - Country of zipcode (Currently service only works for USA)
- FIPS - Fips code
- company_name - If zipcode belongs to a company
- congress_dist - Congressional district served by zip code
- areacode - areacode served by zip code
- timezone - Timezone observed in zip code
- DPBC - DPBC code

Optional Return Fields:

- billable - Indicates whether customer was charged for this query
- cost - Cost of this query

Append247 (Zipcode)

Description: Returns the zipcode for an address
Service Code: AZ
Input Data Fields:

- address - Street address (optional)
- city - City
- state - State

Standard Return Fields:

- streetnum - Street number or number range of address
- predir - Direction indicator before street name
- street - The street name of the address
- street_suffix - Like St, Rd, Pl, etc.
- postdir - Direction indicator after street name
- unit - Unit number or range of street address
- city - Name of city
- state - Name of state
- zip - Zip code
- zip4 - Additional 4 zip code digits
- county - Name of county
- PRUrbCode" - Puerto Rico Urbanization Code
- country - Country of zip code (Currently service only works for USA)
- FIPS - Fips code
- company_name - If zip code belongs to a company
- congress_dist - Congressional district served by zip code
- areacode - areacode served by zip code
- timezone - Timezone observed in zip code
- DPBC - DPBC code

Optional Return Fields:

- billable - Indicates whether customer was charged for this query
- cost - Cost of this query

Balance Inquiry

Description: Returns your current account balance
Service Code: B
Input Data Fields: None
Standard Return Fields:

- balance - Current account balance

Optional Return Fields:

- billable - Indicates whether customer was charged for this query
- cost - Cost of this query

Carrier247 (USA) and Text@

Description: Carrier Lookup for USA/Canada with email-to-sms gateway addresses.
Service Code: CU or MT
Input Data Fields:

- phone - The phone number to query

Standard Return Fields:

- phone - The phone number queried
- wless - Either “y” for wireless or “n” for non-wireless
- carrier_name: carrier name of provided number
- carrier_id:: Carrier id of carrier
- sms_address: SMS gateway address of number
- mms_address: MMS gateway address of number

Optional Return Fields:

- billable - Indicates whether customer was charged for this query
- cost - Cost of this query
- type - Either “M” for mobile phone, “L” for landline, or “V” for VOIP
- GIT - Voicemail greeting interrupt Tone
- ocn - The operating company number
- timezone - The timezone in the area specified by the phone number’s NPA
- tz_offset - The offset from UTC time in the area specified by the numbers NPA
- iana - IANA timezone of the area specified by the number’s NPA
- last_port_date - The date the phone was last ported to a new carrier
- ratecenter - BROKEN
- 'clli' - BROKEN
- assign_date - BROKEN,
- prefix_type - BROKEN
- switch_name - BROKEN,
- switch_type - BROKEN
- lata - BROKEN
- dba - Standard name for carrier
- mno - Mobile Network Operator
- city - City where NPA/NXX is located
- state - State where NPA/NXX is located
- county - Country where NPA/NXX is located
- zip - Zip code where NPA/NXX is located

Carrier247 (International)

Description: Worldwide carrier lookup service
Service Code: CI
Input Data Fields:

- phone - The phone number to query (must include country code)

Standard Return Fields:

- phone - The phone number queried
- wless - Either “y” for wireless or “n” for non-wireless
- carrier_name: carrier name of provided number
- carrier_id:: Carrier id of carrier
- country - The country of the phone number
- isoCountry2 - The ISO 3166 2-digit country code: https://en.wikipedia.org/wiki/ISO_3166-1_alpha-2

Optional Return Fields

- billable - Indicates whether customer was charged for this query
- cost - Cost of this query
- type - Either “M” for mobile phone, “L” for landline, or “V” for VOIP
- GIT - Voicemail greeting interrupt Tone
- sms_address - The email-to-sms gateway address
- mms_address - The email-to-mms gateway address

Carrier247 (Carrier Type)

Description: Returns whether a phone number is wireless, landline or VOIP
Service Code: CT
Input Data Fields:

- phone - The phone number to query (must include country code)

Standard Return Fields:

- phone - The phone number queried
- type - Either “M” for mobile phone, “L” for landline, or “V” for VOIP

Optional Return Fields:

- billable - Indicates whether customer was charged for this query
- cost - Cost of this query

Do-Not-Call (Add)

Description: Adds a phone number to customer’s internal do-not-call list.
Service Code: DA
Input Data Fields:

- phone - Phone number to be added to customer’s internal DNC list

Standard Return Fields:

- phone - Phone number which was added
- action - Will have a value of “ADDED” on success

Optional Return Fields:

- billable - Indicates whether customer was charged for this query
- cost - Cost of this query

Do-Not-Call (Check)

Description: Checks if phone appears on Federal or customer's internal DNC List.
Service Code: DC
Input Data Fields:

- phone - The phone number to check

Standard Return Fields:

- phone - Phone number which was checked
- dnc - Will be one of the following values:
  - OK-TO-DIAL - Phone doesn't appear on Federal or internal list
  - FEDERAL DNC - Phone appears on Federal list
  - INTERNAL DNC - Phone appears on customer's internal DNC list
  - UNAUTHORIZED - Customer doesn't have access to this area code in the Federal DNC list.

Optional Return Fields:

- billable - Indicates whether customer was charged for this query
- cost - Cost of this query

Do-Not-Call (Remove)

Description: Removes a phone number from customer's internal do-not-call list.
Service Code: DR
Input Data Fields:

- phone - Phone number to be removed from customer's internal DNC list

Standard Return Fields:

- phone - Phone number which was removed
- action - Will have a value of "REMOVED" on success

Optional Return Fields:

- billable - Indicates whether customer was charged for this query
- cost - Cost of this query

Locate247 (IP Address)

Description: Finds the geographical location of an AP address
Service Code: GI
Input Data Fields:

- ip - The IP address being queried

Standard Return Fields:

- ip_address - The IP address being queried
- city - The city where the IP address is located
- state - The state where the IP address is located
- country - The country where the IP address is located
- zipcode - The zipcode where the IP address is located

- latitude - The latitude of the IP address
- longitude - The longitude of the IP address
- areacode - The areacode where the IP address is located (if in USA)
- timezone - The timezone where the IP address is located
- dst - Whether DST is active where the IP address is located

Optional Return Fields

- billable - Indicates whether customer was charged for this query
- cost - Cost of this query
- ip_version = IPv4 or IPv6
- is_mobile = If IP belongs to a mobile phone (Y/N)
- phone_carrier_name = Name of phone carrier (if is_mobile)
- phone_carrier_mcc = mcc number for phone carrier
- phone_carrier_mnc = ncc number for phone carrier
- phone_country_code = Country code of phone carrier
- eu_member = true/false
- currency_type = Type of currency
- language = Main Language spoken
- language_code = Language code
- provider_name = Internet provider name
- provider_domain = Internet provider domain name
- provider_type = Type of internet provider
- asn = ASN Number
- route = Route info for Internet connection
- timezone_code = Time zone abbreviation
- is_abuser = IP address is known for abusive activity
- is_attacker = IP address known for malicious activity
- is_bogon = IP address is a "bogon" - an unassigned IP address
- is_cloud_provider = IP address belongs to a cloud service provider
- is_proxy = IP address is a proxy server
- is_relay = IP address is a relay server
- is_tor = IP address is a TOR relay
- is_tor_exit = IP address is a TOR exit relay
- is_vpn = IP address is part of a VPN network
- is_anonymous = IP address is a proxy server, TOR or VPN
- is_threat = IP address is associated with an abuser or and attacker

Trust247 (User Identity)

Description: For mitigating account creation fraud. This service checks the data provided when a user signs-up for an account to determine if the signup is who they say they are.
Service Code: TI
Input Data Fields:

- firstname - The user's first name

- lastname - The user's last name (surname)
- company_name - The name of the user's organization (if applicable)
- company_url - The website for user's organization (if applicable)
- address - The street address supplied by the user
- city - The city supplied by the user
- state - The state supplied by the user
- zip - The postal code supplied by the user
- ip - The IP address the user signed-up from
- email - The email address supplied by the user
- phone - The phone number supplied by the user

Standard Return Fields:

- weight - A 1 to 100 measurement of how likely the signup is to be from a legitimate user. Anything under 50 should generally be considered to be suspicious, and anything over 80 should generally be considered legitimate.
- matches - An array of match statistics:
    - city_ip - If the city and IP address match (true/false)
    - city_phone - If the city matches the phone number (true/false)
    - city_zip - If the city matches the postal code (true/false)
    - company_phone- If the company matches the phone number (true/false)
    - firstname_phone - If the first name matches the phone number (true/false)
    - lastname_phone - If the last name matches the phone number (true/false)
    - state_ip - If the state matches the IP address (true/false)
    - state_phone - If the state matches the phone number (true/false)
    - state_zip - If the state matches the postal code (true/false)
- Notes - An array of notes about the match result. Here's a list of note values:
    - No Phone Provided
    - VOIP Phone Used
    - Invalid Phone Number
    - No Email Address Provided
    - Invalid Email Address Provided
    - Business Email Address Provided
    - Temporary Email Address Used
    - Non-Corporate Email Address Used
    - In-Service Wireless Phone Number Provided
    - Wireless Phone Number Not in Service
    - Non-USA Phone Number
    - Name and Phone Number Match
    - Company and Phone Number Match
    - State-level Phone Match
    - City-level Phone Match
    - No Phone Match
    - No Postal Code Provided
    - State Level Postal Code Match
    - City Level Postal Code Match
    - No Postal Code Match

- Proxy Server
- Probable Proxy Server
- Potential Proxy Server
- State Level IP Address Match
- No Street Address Provided

Optional Return Fields

- billable - Indicates whether customer was charged for this query
- cost - Cost of this query

Verify247 (Email)

Description: Checks if an email address is valid and deliverable
Service Code: VE
Input Data Fields:

- email - The email address to be verified

Standard Return Fields:

- email - The email address which was verified
- valid - YES or NO - whether it's a valid, sendable email address
- reason - Description why/why not address is valid
- free_account - YES/NO - Whether it's a free email account (yahoo, gmail, etc)

Optional Return Fields:

- billable - Indicates whether customer was charged for this query
- cost - Cost of this query
- disposable_email - YES/NO - Whether it's a temporary email address

Verify247 (Phone)

Description: Verifies a phone number is in-service and accepts inbound calls
Service Code: VP
Input Data Fields:

- phone - The phone number to be verified

Standard Return Fields:

- phone - The phone number which was verified
- active - YES/NO - whether phone is active
- confidence - high/low - Confidence level

Optional Return Fields

- billable - Indicates whether customer was charged for this query
- cost - Cost of this query

Note: A phone with an active status of "YES" and a confidence level of "high" is most likely to be an active number, while a phone with an active status of "NO" and a confidence level of "high" is least likely to be an active number.

Verify247 (Postal Address)

Description: Checks a USA postal address and returns corrections to make it fit USPS standards.

Service Code: VA

Input Data Fields:

- address - Street address
- city - City name
- state - State
- zip - Zip code (nnnnn or nnnnn-nnnn)

Standard Return Fields:

- valid - YES/NO - Is address valid?
- addr1 - Line 1 of corrected street address
- addr2 - Line 2 of corrected street address
- streetnum - Street number or number range of address
- predir - Direction indicator before street name
- street - The street name of the address
- street_suffix - Like St, Rd, Pl, etc.
- postdir - Direction indicator after street name
- unit - Unit number or range of street address
- city - Name of city
- state - Name of state
- zip - Zip code
- zip4 - Additional 4 zip code digits
- county - Name of county
- PRUrbCode" - Puerto Rico Urbanization Code
- country - Country of zip code (Currently service only works for USA)
- FIPS - Fips code
- company_name - If zip code belongs to a company
- congress_dist - Congressional district served by zip code
- areacode - areacode served by zip code
- timezone - Timezone observed in zip code
- DPBC - DPBC code

MULTIPLE SERVICES IN A SINGLE API CALL

The Data247 API allows multiple data services to be used in a single API call. For example, someone may want to use both Phone and Email append services to get a consumer's phone and their email address. The resulting query would look like this:

https://api.data247.com/v3.0?key=YOURAPIKEY&api=AP,AE&fname=Joe&lname=Smith&address=12 Main Street&city=Nashua&state=NH

The result for this query would contain all of the return fields from the AP query and all of the return fields from the AE query. Any return fields which are returned by both services will only appear once in the return data.

Here's a more interesting example:

https://api.data247.com/v3.0?key=YOURAPIKEY&api=AN,AG&phone=16175551212

The AN service requires a phone number and returns a name. The AG service requires a name and returns a gender. But only the phone number is being passed in. The API is smart enough to know that the name required by the AG service can be obtained from the result of the AN service.

MULTIPLE QUERIES IN A SINGLE API CALL

The "API Request Format" section above explains how to send one query at a time using the API, but the API is capable of sending multiple queries at once. Depending on the service type, there are one or two ways to do this:

MULTIPLE PHONE NUMBER PARAMETERS

This method only applies to the following services:

- Text@
- Carrier247 (USA)
- Carrier247 (International)
- Carrier247 (Carrier Type)
- Do-Not-Call
- Append247 (Reverse Phone)

For the above services, one can submit multiple phone numbers by listing them in the URL as p1=<phone>&p2=<phone>&p3=<phone> . . .

Using this method, up to 99 phone numbers may be specified in a single API call. Here's an example:

https://api.data247.com/v3.0?key=MyAPIKey&api=DC&p1=15085551212&p2=17815551212&p3=16035551212&p4=16175551212

POSTING A JSON DOCUMENT

This second method works with all API services. You may call the API as an HTTPS POST, and instead of adding the input data as URL parameters, they may instead be POSTed as a JSON-format document, as follows:

```
{
   "XX": [
        {
           "query 1 parameter": "value",
           "query 1 parameter": "value",
           "query 1 parameter": "value",
           "query 1 parameter": "value"
        },
        {
           "query 2 parameter": "value",
           "query 2 parameter": "value",
           "query 2 parameter": "value",
           "query 2 parameter": "value"
        },
           "query 3 parameter": "value",
           "query 3 parameter": "value",
           "query 3 parameter": "value",
           "query 3 parameter": "value"
     ]
}
```

. . . where "XX" is the service code.

The API will allow up to 2,500 queries in a single JSON document, but note that some Data247 services are slower than others, and for some it could cause a timeout to occur.

Here's an example JSON which could be POSTed to the API:

```
{
    "AP": [
        {
            "firstname": "Joe",
            "lastname": "Smith",
            "address": "12 Main Street",
            "city": "Dallas",
            "state": "TX",
            "zip": "75201"
        },
        {
            "firstname": "Sarah",
            "lastname": "Jones",
            "address": "10 Elm Street",
            "city": "Fitchburg",
            "state": "MA",
            "zip": "01420"
        }
    ]
}
```

API RESPONSE FORMAT FOR MULTIPLE QUERIES

When multiple queries are sent in a single API call, the API response will differ in two ways:

1. The array of "results" in the response will contain multiple items instead of just one.

2. In addition to the "status" indicator at the response level, there will also be a "status" indicator present in the result set for each query. This query-level status will indicate the status of that particular query. A status of "OK" means the query was successful. Otherwise the status will contain an error message (see API ERROR MESSAGES section below)

Here's an example of a response for multiple queries in JSON format:

```
{
  response :{
   status : "OK",
   results :[
              {
                status : "OK",
                phone : "15084041234",
                wless : "n",
                carrier_name : "Bandwidth.com",
                carrier_id : 50025,
                sms_address : "",
                mms_address : ""
              },
              {
                status : "OK",
                phone : "17818640001",
                wless : "y",
                carrier_name : "Bellsouth Mobility, LLC (AT+T)",
                carrier_id : 6,
                sms_address : "7818640001@txt.att.net",
                mms_address : "7818640001@mms.att.net"
              }
            ]
            }
}
```

API ERROR MESSAGES

ERROR MESSAGE	MEANING
D247_INVALID_CREDENTIALS	Wrong key value provided
D247_INVALID_API_KEY	The API key being used is invalid.
D247_INVALID_SRC_IP	IP Authentication is turned on for your account, but the source IP address is not in the list of allowed addresses.
D247_ACCOUNT_INACTIVE	Account set to inactive, possible reasons could be expired subscription, no funds left on the account
D247_INSUFFICIENT_FUNDS	Not enough funds left on the account.
D247_MISSING_REQUIRED_PARAMS	Not all required parameters for specific api is provided or provided parameters are invalid
D247_PROCESSING_ERROR	Some unknown error occurred while processing your request
D247_INVALID_API	Wrong api parameter value provided
D247_NO_DATA_FOUND	No related data found on our database
D247_MISSING_PHONE_NUM	Data service you requested requires a phone number, but you didn't provide it
D247_INVALID_PHONE	The phone number provided is not valid
D247_INVALID_EMAIL	The email address provided is not valid
D247_CANNOT_ADD_NUMBER	The phone number you provided cannot be added to DNC list
D247_CANNOT_REMOVE_NUMBER	The phone number you provided cannot be removed from a DNC list
D247_INVALID_ZIPCODE	The zip code specified does not exist
D247_NO_SUBCLIENT_DATA	Not data exists for this subclient
D247_INVALID_IP_ADDRESS	The IP address is invalid
D247_ADDRESS_NOT_FOUND	The postal address was not found in our databsase
D247_INVALID_ADDRESS	The postal address is not valid

D247_DATA_RESTRICTED	You do not have access to this data.
D247_NOT_IMPLEMENTED	This feature has not been implemented yet
D247_NO_CANADA_ACCESS	You do not have access to Canadian data
D247_TOO_MANY_QUERIES	Too many queries in this API call
D247_NOT_AUTHORIZED	Contact Data247 customer support to be granted access.

EXAMPLES IN VARIOUS PROGRAMMING LANGUAGES

Axios / Angular

```
var axios = require('axios');
// for angular use below
// import axios from axios

var config = {
  method: 'get',
  url: 'https://api.data247.com/v3.0?key=YourAPIKey&api=MT&p1=YouPhoneNumber',
  headers: { }
};

axios(config)
.then(function (response) {
  let resp = JSON.parse(response)
  let carrier = resp.carrier_name
  let is_wireless = resp.wless
  let SMS_gateway = resp.sms_address
  console.log();
})
.catch(function (error) {
console.log(error);
});
```

C

```
#include <stdio.h>
#include <stdlib.h>
#include <curl/curl.h>

int main(int argc, char *argv[]) {

  CURL *curl;
  CURLcode res;
  char APIKEY[] = "AddYourAPIKeyHere";
  char url[] = "https://api.data247.com/v3.0?";
  char phone[] = "17818640001";
  char postData[255];

  sprintf(postData,"key=%s&api=MT&p1=%si&out=yaml",APIKEY,phone);
  curl = curl_easy_init();
  if(curl) {
    curl_easy_setopt(curl, CURLOPT_URL, url);
    curl_easy_setopt(curl, CURLOPT_POSTFIELDS, postData);
    res = curl_easy_perform(curl);
    curl_easy_cleanup(curl);
  }
  printf("%s",res);
  return 0;
}
```

C#

```
using System;
using System.Net;
using Newtonsoft.Json.Linq;

class Program
{
    static void Main()
    {
        string key = "YourAPIKey";
        string phone = "7818641234";

        try
        {
            WebClient client = new WebClient();
            string response =
client.DownloadString("https://api.data247.com/v3.0?key=" + key + "&api=MT&p1=" +
phone);
            var parsedJson = JObject.Parse(response);
            Console.Write("Phone number:{0} \n",
parsedJson.SelectToken("response.results[0].phone"));
            Console.Write("Carrier Name:{0} \n",
parsedJson.SelectToken("response.results[0].carrier_name"));
            Console.Write("Is wireless :{0} \n",
parsedJson.SelectToken("response.results[0].wless"));
            Console.Write("SMS gateway :{0} \n",
parsedJson.SelectToken("response.results[0].sms_address"));
        }
        catch(Exception e)
        {
            Console.Write("Error : {0}", e);
        }
    }
}
```

cURL

single query:

```
curl 'https://api.data247.com/v3.0?key=YourAPIKey&api=MT&phone=16175551212'
```

multiple queries in a single API call:

```
curl -i -X POST -H "Content-Type:application/json" \
   -d '{
       "CU": [
                    {
                       "phone": "15084041234"
                    },
                    {
                       "phone": "17818640001"
                    }
              ]
        }' 'https://api.data247.com/v3.0?key=YourAPIKey&api=CU'
```

Go

```
package main

import (
  "fmt"
  "net/http"
  "io/ioutil"
)

type Input struct {
  carrier_name string `json:"carrier_name"`
  wless string `json:"wless"`
  sms_address int8 `json:"sms_address"`
}

var input Input
func main() {

  url := "https://api.data247.com/v3.0?key=YourAPIKey&api=MT&p1=YouPhoneNumber"
  method := "GET"

  client := &http.Client {
  }
  req, err := http.NewRequest(method, url, nil)

  if err != nil {
    fmt.Println(err)
    return
  }
  res, err := client.Do(req)
  if err != nil {
    fmt.Println(err)
    return
  }
  defer res.Body.Close()

  body, err := ioutil.ReadAll(res.Body)
  err := json.NewDecoder(r.Body).Decode(&input)
  if err != nil {
    fmt.Println(err)
    return
```

```
  }
  fmt.Println(string(body))
  fmt.Fprintf(w, "carrier_name: %s", input.carrier_name)
  fmt.Fprintf(w, "wless: %s", input.wless)
  fmt.Fprintf(w, "sms_address: %s", input.sms_address)
}
```

Java

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;

public class test {
    public static void main(String[] args) {
        try {
            String apikey = "YourAPIKey";
            URL url = new URL("https://api.data247.com/v3.0?key=" + apikey +
"&api=MT&phone=17815551212");
            HttpURLConnection conn = (HttpURLConnection) url.openConnection();
            conn.setRequestMethod("GET");
            conn.setRequestProperty("Accept", "application/json");
            if (conn.getResponseCode() != 200) {
              throw new RuntimeException("Failed : HTTP Error code : " +
conn.getResponseCode());
            }
            InputStreamReader in = new InputStreamReader(conn.getInputStream());
            BufferedReader br = new BufferedReader(in);
            String output = br.readLine();
            System.out.println("Output = " + output);
            conn.disconnect();

        } catch (Exception e) {
            System.out.println("Exception in example:- " + e);
        }
    }
}
```

Node.js

```
const request = require('request');

const API_KEY = "YourAPIKey";
const phone   = "17818640001";

request("https://api.data247.com/v3.0?key=" + API_KEY + "&api=MT&p1=" + phone, {
json: true }, (err, res, body) => {
  if (err) { return console.log(err); }
  res = body.response.results[0];
```

```
  console.log("Carrier Name: " + res.carrier_name);
  console.log("Is Wireless : " + res.wless);
  console.log("sms_gateway : " + res.sms_address);
  console.log("mms_gateway : " + res.mms_address);
});
```

Perl

```
use REST::Client;

my $client = REST::Client->new();

$content = {'Content-type' => 'application/json'};
$url="http://api.data247.com/v3.0";
$req = '{
          "key"   : "YourAPIKey",
          "api"   : "MT",
          "phone" : "17818640108"
        }';
$client->POST($url,$req,$content);
print $client->responseContent();
```

PHP

```
<?php

  $key = "YourAPIKey";  // Replace key value with your own api key
  $phone = "17818641234";

  $url = "https://api.data247.com/v3.0?key=$key&api=MT&p1=$phone";
  $result = @file_get_contents($url);
  if ($result){
    $result = @json_decode($result, true);
    if (!empty($result['response']['status']) && $result['response']['status'] == 'OK'){
      $response = $result['response']['results'][0];
      echo "Phone      : $phone\n";
      echo "carrier    : " . $response['carrier_name'] . "\n";
      echo "Is wireless: " . $response['wless'] . "\n";
      echo "SMS gateway: " . $response['sms_address'] . "\n";
    } else {
      echo "Error: " . $result['response']['status'] . "\n";
    }
  } else {
    echo "Error calling API\n";
  }
?>
```

Python

```
import requests

key   = "YourAPIKey"
phone = "7818641234"

try:
  r = requests.get("https://api.data247.com/v3.0?key=%s&api=MT&p1=%s" %
(key,phone))
  adict = r.json()['response']
  if 'status' in adict and adict['status'] == 'OK':
    result = adict['results'][0]
    print("Phone number: %s" % (phone))
    print("Carrier Name: %s" % (result['carrier_name']))
    print("Is wireless : %s" % (result['wless']))
    print("SMS gateway : %s" % (result['sms_address']))
  else:
    print("Error: %s \n" % (adict['status']))
except Exception as err:
  print('Error: '+ str(err))
```

Ruby

```
require 'httparty'

apikey = "YourAPIKey"
phone  = "17818640001"

response =
HTTParty.get("https://api.data247.com/v3.0?key=#{apikey}&api=MT&phone=#{phone}"
)
js = JSON.parse(response.body)['response']
if (js['status'] == "OK") then
  data = js['results'][0]
  print "Carrier Name = #{data['carrier_name']}\n"
  print "Is Wireless  = #{data['wless']}\n"
  print "SMS Gateway  = #{data['sms_address']}\n"
  print "MMS Gateway  = #{data['mms_address']}\n"
else
  print "Error: #{js['status']}\n"
end
```

VB.Net

```
Imports System
Imports System.Net
Imports Newtonsoft.Json.Linq

Class Program
  Private Shared Sub Main()
    Dim key As String = "YourAPIKey"
    Dim phone As String = "7818641234"

    Try
      Dim client As WebClient = New WebClient()
      Dim response As String = client.DownloadString("https://api.data247.com/v3.0?key=" & key & "&api=MT&p1=" & phone)
      Dim parsedJson = JObject.Parse(response)
      Console.Write("Phone number:{0} " & vbLf, parsedJson.SelectToken("response.results[0].phone"))
      Console.Write("Carrier Name:{0} " & vbLf, parsedJson.SelectToken("response.results[0].carrier_name"))
      Console.Write("Is wireless :{0} " & vbLf, parsedJson.SelectToken("response.results[0].wless"))
      Console.Write("SMS gateway :{0} " & vbLf, parsedJson.SelectToken("response.results[0].sms_address"))
    Catch e As Exception
      Console.Write("Error : {0}", e)
    End Try
  End Sub
End Class
```

MIGRATING FROM DATA247 API VERSION 2.0 to 3.0

If you're currently using version 2 of our API, this section will help you migrate to version 3.

Visual Comparison of API Calls

Example Version 2.0 API call:

https://api.data24-7.com/v/2.0?user=<USERNAME>&pass=<PASSWORD>&api=<V2_SVC_CODE>&p1=<PHONE>

Example Version 3.0 API call:

https://api.data247.com/v3.0?key=<API_KEY>&api=<V3_SVC_CODE>&p1=<PHONE>&out=xml

Steps to upgrade from version 2.0 to 3.0

STEP 1: Generate an API key

Follow these instructions to generate a new API key:

a. Log into your Data247 account
b. From the "Account" menu, choose "API Key / Password"
c. Click on the "API Version 3" tab.
d. Enter a description in the "Key Description" box.
e. Click the "Generate API Key" button.
f. Highlight and copy your new API key.

STEP 2: Change the Domain

Change the domain part of the URL from https://api.data24-7.com/v/2.0? to https://api.data247.com/v3.0?. Note the removal of the dash in the domain name.

STEP 3: Use API key

Remove the username and password part of the URL and replace it with key=<API_KEY>, where <API_KEY> is replaced with the API key you generated in step 1 above.

STEP 4: Update Service Code

The API service codes have changed from v2 to v3. So replace the letter(s) after api= as described in the chart below:

V2_SERVICE	V2_SERVICE_CODE	V3_SERVICE_CODE

Text@	T	MT
Carrier24-7	C	CI
Email Verification	E	VE
Phone Append	XP	AP
Email Append	XE	AE
Rev. Phone Append	A	APR
Profile24-7	F	AR
CNAM / Name Append	I	AN
Zip24-7	Z	AZR
IP24-7	P	GI
Do-Not-Call	DC	DC

STEP 5 : Set Output Format to XML

API version 2.0 defaulted to XML as the output format, and version 3.0 defaults to JSON. If you want your API results to remain in XML format, add &out=xml to the URL.

STEP 6: Update Code to Handle Differences in Response:

The outputs returned by both API versions are very similar, but there are minor differences, including the positioning of the <status> indicator, and some field name changes. See the example below:

TEXT@ VERSION 2.0 RESPONSE:

```
<response>
  <results>
    <result item="1">
      <status>OK</status>
      <number>17818640111</number>
      <wless>y</wless>
      <carrier_name>Sprint Spectrum, L.P.</carrier_name>
      <carrier_id>9</carrier_id>
      <sms_address>7818640111@messaging.sprintpcs.com</sms_address>
      <mms_address>7818640111@pm.sprint.com</mms_address>
    </result>
  </results>
</response>
```

TEXT@ VERSION 3.0 RESPONSE:

```
<response>
  <status>OK</status>
  <results>
    <result>
      <phone>17818640111</phone>
      <wless>y</wless>
      <carrier_name>Sprint Spectrum, L.P.</carrier_name>
      <carrier_id>9</carrier_id>
      <sms_address>7818640111@messaging.sprintpcs.com</sms_address>
      <mms_address>7818640111@pm.sprint.com</mms_address>
    </result>
  </results>
</response>
```