Address Capture Quick Start Guide

Start using Address Capture in three steps. Set up your API key, search for addresses with Find, and retrieve full details with Retrieve.

Address Capture provides real-time address search and verification as users type, reducing errors and improving data quality.

Coverage

- **Global:** Address data for over 250 countries and territories
- **UK Premium:** AddressBase Premium data with Geolocation support

Available Endpoints

Address Capture provides three API endpoints:

Core Endpoints

- **Find** - Search for addresses and places as users type
- **Retrieve** - Get full address details based on the selected ID

Additional Endpoint

- **Geolocation** - Find addresses within a radius of latitude/longitude coordinates (UK AddressBase Premium only)

Getting Started

1. **Create an API key** if you don't have one
2. Make a **Find** request to search for addresses as the user types
3. Make a **Retrieve** request to get the full address when they select one

Here's a simple example request for the **Find** endpoint:

cURL

```
curl --request GET \
  --url https://api.addressy.com/Capture/Interactive/Find/v1.20/json6.ws
```

Python

```
import requests

url = "https://api.addressy.com/Capture/Interactive/Find/v1.20/json6.ws"

response = requests.get(url)

print(response.json())
```

JavaScript

```
const url = 'https://api.addressy.com/Capture/Interactive/Find/v1.20/json6.ws';
const options = {method: 'GET', body: undefined};

try {
  const response = await fetch(url, options);
  const data = await response.json();
  console.log(data);
} catch (error) {
  console.error(error);
}
```

PHP

```php
<?php

$curl = curl_init();

curl_setopt_array($curl, [
  CURLOPT_URL => "https://api.addressy.com/Capture/Interactive/Find/v1.20/json6.w
  CURLOPT_RETURNTRANSFER => true,
  CURLOPT_ENCODING => "",
  CURLOPT_MAXREDIRS => 10,
  CURLOPT_TIMEOUT => 30,
  CURLOPT_HTTP_VERSION => CURL_HTTP_VERSION_1_1,
  CURLOPT_CUSTOMREQUEST => "GET",
]);

$response = curl_exec($curl);
$err = curl_error($curl);

curl_close($curl);

if ($err) {
  echo "cURL Error #:" . $err;
} else {
  echo $response;
}
```

Go

```go
package main

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

func main() {

    url := "https://api.addressy.com/Capture/Interactive/Find/v1.20/json6.ws"

    req, _ := http.NewRequest("GET", url, nil)

    res, _ := http.DefaultClient.Do(req)

    defer res.Body.Close()
    body, _ := io.ReadAll(res.Body)

    fmt.Println(res)
    fmt.Println(string(body))

}
```

Java

```java
HttpResponse<String> response = Unirest.get("https://api.addressy.com/Capture/Int
  .asString();
```

Ruby

```ruby
require 'uri'
require 'net/http'

url = URI("https://api.addressy.com/Capture/Interactive/Find/v1.20/json6.ws")

http = Net::HTTP.new(url.host, url.port)
http.use_ssl = true

request = Net::HTTP::Get.new(url)

response = http.request(request)
puts response.read_body
```

Here's a simple example request for the **Retrieve** endpoint:

cURL

```curl
curl --request GET \
  --url https://api.addressy.com/Capture/Interactive/Retrieve/v1.30/json6.ws
```

Python

```python
import requests

url = "https://api.addressy.com/Capture/Interactive/Retrieve/v1.30/json6.ws"

response = requests.get(url)

print(response.json())
```

JavaScript

```javascript
const url = 'https://api.addressy.com/Capture/Interactive/Retrieve/v1.30/json6.ws
const options = {method: 'GET', body: undefined};

try {
  const response = await fetch(url, options);
  const data = await response.json();
  console.log(data);
} catch (error) {
  console.error(error);
}
```

PHP

```php
<?php

$curl = curl_init();

curl_setopt_array($curl, [
  CURLOPT_URL => "https://api.addressy.com/Capture/Interactive/Retrieve/v1.30/jso
  CURLOPT_RETURNTRANSFER => true,
  CURLOPT_ENCODING => "",
  CURLOPT_MAXREDIRS => 10,
  CURLOPT_TIMEOUT => 30,
  CURLOPT_HTTP_VERSION => CURL_HTTP_VERSION_1_1,
  CURLOPT_CUSTOMREQUEST => "GET",
]);

$response = curl_exec($curl);
$err = curl_error($curl);

curl_close($curl);

if ($err) {
  echo "cURL Error #:" . $err;
} else {
  echo $response;
}
```

Go

```go
package main

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

func main() {

    url := "https://api.addressy.com/Capture/Interactive/Retrieve/v1.30/json6.ws'

    req, _ := http.NewRequest("GET", url, nil)

    res, _ := http.DefaultClient.Do(req)

    defer res.Body.Close()
    body, _ := io.ReadAll(res.Body)

    fmt.Println(res)
    fmt.Println(string(body))

}
```

Java

```java
HttpResponse<String> response = Unirest.get("https://api.addressy.com/Capture/Int
  .asString();
```

Ruby

```ruby
require 'uri'
require 'net/http'

url = URI("https://api.addressy.com/Capture/Interactive/Retrieve/v1.30/json6.ws")

http = Net::HTTP.new(url.host, url.port)
http.use_ssl = true

request = Net::HTTP::Get.new(url)

response = http.request(request)
puts response.read_body
```

What Address Capture Returns

A typical Address Capture workflow:

- **Find** returns a list of matching addresses based on search text
- **Retrieve** returns the complete formatted address in multiple formats

Find Response

Response should be in the following format:


```
{
  "Items": [
    {
      "Id": "GB|RM|B|55605138|ENG",
      "Type": "Address",
      "Text": "G B G Evermore 128 Queen Victoria Street",
      "Highlight": "0-1,2-3,4-5",
      "Description": "London EC4V 4BJ"
    }
  ]
}
```


Retrieve Response

```
{
  "Items": [
    {
      "Id": "GB|RM|B|55605138|ENG",
      "DomesticId": "55605138",
      "Language": "ENG",
      "LanguageAlternatives": "",
      "Department": "",
      "Company": "G B G",
      "SubBuilding": "",
      "BuildingNumber": "128",
      "BuildingName": "Evermore",
      "SecondaryStreet": "",
      "Street": "Queen Victoria Street",
      "Block": "",
      "Neighbourhood": "",
      "District": "",
      "City": "London",
      "Line1": "G B G Evermore",
      "Line2": "128 Queen Victoria Street",
      "Line3": "",
      "Line4": "",
      "Line5": "",
      "AdminAreaName": "",
      "AdminAreaCode": "",
      "Province": "",
      "ProvinceName": "",
      "ProvinceCode": "",
      "PostalCode": "EC4V 4BJ",
      "CountryName": "United Kingdom",
      "CountryIso2": "GB",
      "CountryIso3": "GBR",
      "CountryIsoNumber": "826",
      "SortingNumber1": "",
      "SortingNumber2": "",
      "Barcode": "",
      "POBoxNumber": "",
      "Label": "G B G Evermore\n128 Queen Victoria Street\nLondon\nEC4V 4BJ\nUnit
      "Type": "Commercial",
      "DataLevel": "Premise",
      "Field1": "",
      "Field2": "",
```

```
      "Field3": "",
      "Field4": "",
      "Field5": "",
      "Field6": "",
      "Field7": "",
      "Field8": "",
      "Field9": "",
      "Field10": "",
      "Field11": "",
      "Field12": "",
      "Field13": "",
      "Field14": "",
      "Field15": "",
      "Field16": "",
      "Field17": "",
      "Field18": "",
      "Field19": "",
      "Field20": ""
    }
  ]
}
```


And you can use this information below to interpret the response fields:

▼ Find Response Fields

Search Results

Field	Type	Description
`Id`	string	Unique identifier for the address or container (for further results). Can change over time
`Type`	string	Type of result: Address, Container, Postcode, Street, BuildingName, or Building. If Type is Address, pass Id to Retrieve. Otherwise, pass Id as Container to another Find request

Field	Type	Description
`Text`	string	First half of the address displayed to the user
`Highlight`	string	Number ranges showing which characters match the search term. Use to highlight matching text in bold. Empty if Container Id included in request
`Description`	string	Second half of the address, including address count for Containers

▼ Retrieve Response Fields

Address Label Format

Field	Type	Description
`Label`	string	Complete formatted address with line breaks (\n). Ideal for display or printing on envelopes

Address Lines Format (Recommended)

Field	Type	Description
`Company`	string	Company name
`Line1`	string	First line of formatted address
`Line2`	string	Second line of formatted address
`Line3`	string	Third line of formatted address
`Line4`	string	Fourth line of formatted address
`Line5`	string	Fifth line of formatted address
`City`	string	Town or city name

Field	Type	Description
`Province`	string	Either ProvinceName or ProvinceCode based on country standards
`PostalCode`	string	Full postal code (Postcode, ZIP Code, etc.)

Address Elements Format

Field	Type	Description
`Department`	string	Department name for business addresses
`SubBuilding`	string	Sub-building identifier (Flat 4, Unit 2, etc.)
`BuildingNumber`	string	House or building number
`BuildingName`	string	Building name if applicable
`SecondaryStreet`	string	Secondary thoroughfare name
`Street`	string	Main thoroughfare name
`Block`	string	Block details
`Neighbourhood`	string	Neighbourhood name
`District`	string	District name

Geographic and Administrative

Field	Type	Description
`AdminAreaName`	string	Administrative area name (e.g., county council)
`AdminAreaCode`	string	Administrative area code
`ProvinceName`	string	Largest administrative division name (state, county)
`ProvinceCode`	string	Administrative division code

Field	Type	Description
`CountryName`	string	Full country name (ISO 3166-1)
`CountryIso2`	string	Two-character ISO country code
`CountryIso3`	string	Three-character ISO country code
`CountryIsoNumber`	string	ISO numeric country code

Additional Fields

Field	Type	Description
`Id`	string	Unique address identifier
`DomesticId`	string	Reference number in source dataset (e.g., UDPRN)
`Language`	string	Three-letter language code
`Type`	string	Address type: Residential, Commercial, or Unknown (available in select countries)
`DataLevel`	string	Data precision level: Unknown, Premise, RangedPremise, Street, or City
`POBoxNumber`	string	PO Box number if applicable
`Barcode`	string	Barcode for bulk mailing schemes
`Field1` - `Field20`	string	Custom fields when using Field1Format - Field20Format parameters

Note: Use Field1Format through Field20Format request parameters to return custom data (e.g., latitude, longitude) in these fields.

Error Response Fields

When an error occurs, the API returns an error response with the following structure:

Field	Type	Description
`Error`	string	The error ID
`Description`	string	A description of the error
`Cause`	string	The cause of the error
`Resolution`	string	Actions to resolve the error

Common Error Codes

Status Code	Meaning	Common Causes
400	Bad Request	Invalid parameters, malformed request, missing required fields
401	Unauthorized	Invalid or missing API key
403	Forbidden	API key doesn't have permission for this service, or daily limit exceeded
500	Internal Server Error	Server-side issue, temporary service disruption

Tip: Check the `Resolution` field for specific steps to fix the error. For more details on error responses, see **Find** in API documentation.

Note: Address Capture provides three response formats: Label (single string), Lines (recommended for forms), and Elements (individual components). Use one format consistently to avoid duplication or missing data.

Integration Methods

Beyond direct API integration, Address Capture can be integrated using:

- **Tag:** Automatically-generated code snippets for your website forms - **Tag Setup Guide**
- **SDK:** Pre-built SDK for quick custom integration - **Capture SDK**
- **Pre-built integrations:** Ready-made integrations for Adobe Commerce, Shopify Plus, and more - **All Integrations**

See the **Ways to Integrate** page for guidance on choosing the best method.

Try Address Capture

Test Address Capture using the **Find Playground** to make your first request.

FAQ

Which integration method works best for me?

The best method depends on your technical requirements and resources. Use Tags for quick website integration, the SDK for custom web applications, direct APIs for full control, or pre-built integrations for supported platforms. See the **Ways to Integrate** page for detailed guidance.

How do I map Address Capture fields to my form?

Address Capture provides three formats: **Label** (single formatted string), **Lines** (recommended for multi-line forms), and **Elements** (individual address components). Choose one format based on your form structure. The Lines format works best for most forms with separate fields for company, address lines, city, and postcode. Do not mix formats to avoid duplication or missing data. See the **Retrieve API documentation** for all available fields.

Should I make API calls client-side or server-side?

Consider performance, complexity, and security when deciding. Client-side calls provide faster user experience but expose your API key. Server-side calls are more secure but add latency. See the **API Security** page for detailed guidance on integration options and security best practices.

Can I get latitude and longitude coordinates?

Yes. Use the `field1format={Latitude}` and `field2format={Longitude}` parameters in your Retrieve request to return coordinates in Field1 and Field2. Coordinates are returned where available in the dataset. For Tag integrations,

configure these parameters in the mappings wizard. See the **Retrieve API documentation** for details on custom field formatting.