Articles on: Property Search & Filtering

Searching, Filtering, and Pagination Guide

Our real estate APIs provide powerful and flexible search capabilities, allowing you to find the perfect listings using a variety of query string parameters. In this article, we'll guide you through the process of searching, filtering, and paginating listings using our API.

Getting Started

To search for real estate listings, you will make a GET request to our listings endpoint. By appending query string parameters to the URL, you can filter the listings to match your specific criteria.

Example Search Request

Let's say you want to find homes in New York that have three bedrooms and are priced under one million dollars. You would make a request like this:

```
GET https://api.repliers.io/listings?maxPrice=1000000&city=New York&minBedrooms=3
```

This request will return an array of listings that match the search criteria you specified.

Supported Filters

Our API supports a wide range of filters to help you narrow down your search. Here are some of the most commonly used filters:

maxPrice: The maximum price of the listings.
minPrice: The minimum price of the listings.
city: The city where the listings are located.

minBedrooms: The minimum number of bedrooms.
maxBedrooms: The maximum number of bedrooms.
minBaths: The minimum number of bathrooms.
maxBaths: The maximum number of bathrooms.
propertyType: The type of property (e.g., house, condo, townhouse).
status: The status of the listing (e.g., active, pending, sold).
minSqft: The minimum square footage of the property.
maxSqft: The maximum square footage of the property.

For a complete list of supported filters, please refer to our API reference.

Pagination

When dealing with large datasets, it's essential to use pagination to manage and navigate through the data efficiently. Our API provides pagination details in the response to help you handle this effectively.

Example Response with Pagination Details

Here’s an example of a response that includes pagination information:


```
{
   "page": 1,
   "numPages": 802,
   "pageSize": 100,
   "count": 80198,
   "listings": [
      {
         "mlsNumber": "W9012677",
         "resource": "Property:2381",
         "status": "A",
         "class": "CommercialProperty",
         "type": "Lease",
         "listPrice": "1.00",
         ...
```

```
    },
    ...
  ]
}
```

Pagination Parameters

Our API supports several pagination parameters to control the data returned in each request:

page: The current page of results you are viewing.
resultsPerPage: The number of listings returned per page.
numPages: The total number of pages available.
count: The total number of listings available that match the search criteria.

Using Pagination Parameters

When constructing your search query, you can include pagination parameters to specify which page of results you want to view and how many results per page you want to retrieve.

Example 1: Viewing the Second Page of Results

To view the second page of results with a page size of 50 listings per page, you would use the following request:

```
GET https://api.repliers.io/listings?maxPrice=1000000&city=New York&minBedrooms=3&pageNum=2&resultsPerPage=50
```

This request will return the second page of listings, with each page containing up to 50 listings.

Handling the Response

The response from the API will be a JSON object containing an array of listings that match your search criteria, along with pagination details. Each listing will include detailed information such as the address, price, number of bedrooms and bathrooms, property type, and more.

Example Response


```
{
   "page": 2,
   "numPages": 1603,
   "pageSize": 50,
   "count": 80198,
   "listings": [
      {
         "mlsNumber": "W9012677",
         "resource": "Property:2381",
         "status": "A",
         "class": "CommercialProperty",
         "type": "Lease",
         "listPrice": "1.00",
         "address": "123 Main St, New York, NY",
         "price": 950000,
         "beds": 3,
         "baths": 2,
         "propertyType": "house",
         "status": "active",
         "sqFt": 2000,
         "listingDate": "2023-06-15"
      },
      {
         "mlsNumber": "W9012678",
         "resource": "Property:2382",
         "status": "A",
         "class": "Residential",
         "type": "Sale",
         "listPrice": "980000.00",
         "address": "456 Elm St, New York, NY",
         "price": 980000,
         "beds": 3,
```

```
        "baths": 3,
        "propertyType": "house",
        "status": "active",
        "sqFt": 2100,
        "listingDate": "2023-06-18"
      }...
  ]
}
```


Searching, filtering, and paginating listings with Repliers' real estate APIs is straightforward and flexible, allowing you to tailor your queries to meet your specific needs. For more detailed information on all the available filters and pagination parameters, please visit our API reference.

What questions does this article answer?

How do I search real estate listings using the Repliers /listings endpoint?
How do I pass query string parameters (price, beds, property type, etc.) to filter listings?
How do I implement pagination (page/limit/offset) with the Repliers Listings API?
How should I structure listing search requests for typical real estate UIs (search forms, filters, etc.)?
How do I combine multiple filters in one request and understand how they interact?

Articles on: Property Search & Filtering
Utilizing AI-Powered NLP for Real Estate Listing Searches
Our real estate APIs include an advanced AI-powered Natural Language Processing (NLP) feature that enables you to collect natural language prompts from your users and seamlessly convert them into search API requests. This cutting-edge functionality is rapidly becoming essential in real estate listing searches, particularly for chatbot and voice applications.

How It Works

The NLP feature processes natural language prompts provided by your users and generates the corresponding search API requests. This functionality not only enhances user experience but also streamlines the process of finding relevant listings based on specific criteria.

Example Scenario

Let's say a user inputs the following prompt:

Find me a condo in Toronto with at least 1 bedroom in the Annex. It must have underground parking.

By sending this prompt to the NLP endpoint, you can obtain a formatted API request ready for use.

Request:

POST https://api.repliers.io/nlp

```
{
   "prompt": "Find me a condo in toronto with at least 1 bedroom in the annex. It must have underground parking."
}
```

Response:

```
{
   "request": {
      "url": "https://api.repliers.io/listings?area=Toronto&city=Toronto&propertyType=Condo%20Apt&minBeds=1&neighborhood=Annex&garage=Underground",
      "body": null,
      "summary": "Searching for a condominium apartment in Toronto with a minimum of 1 bedroom located in the Annex neighborhood. The property must have underground parking."
   },
   "nlpId": "12345"
}
```

In this example, the NLP feature has converted the user's natural language input into a structured search request, which can be sent directly to our Listings API to retrieve relevant property listings.

Context-Aware NLP with NLP IDs

Our system is designed to support context-aware interactions using NLP IDs, enabling a more conversational and intuitive search experience. When you send a new prompt to our NLP tool, the system returns an NLP ID in the response. This ID can be included in subsequent prompts to maintain context, allowing the NLP tool to consider both the original and new prompts when generating the API request.

Prompt 1:

I'm looking for a condo to buy with 4 bedrooms.

Request:

POST https://api.repliers.io/nlp

```
{
   "prompt": "I'm looking for a condo to buy with 4 bedrooms.",
}
```

Response:

```
{
   "request": {
      "url": "https://api.repliers.io/listings?class=condo&minBeds=4&type=sale",
      "body": null,
      "summary": "Searching for a condo to buy with a minimum of 4 bedrooms."
   },
   "nlpId": "12345"
```

}

Prompt 2:

Also, I forgot, my budget is 500k, and I need at least 1 parking spot.

Request:

POST https://api.repliers.io/nlp


```
{
   "nlpId": "12345"
   "prompt": "Also, I forgot, my budget is 500k, and I need at least 1 parking spot."
}
```


Response:


```
{
   "request": {
        "url":
"https://api.repliers.io/listings?class=condo&minBeds=4&type=sale&minParkingSpaces=1&max
Price=500000",
        "body": null,
        "summary": "Searching for a Condo Apt to buy with at least 4 bedrooms, within a budget of
500,000, and at least 1 parking spot."
    },
   "nlpId": "12345"
}
```


In this example, the initial request (Prompt 1) generates an nlpId which can be used in subsequent requests to maintain context. When the user adds new criteria or changes the existing prompt, the nlpId is passed along with the new prompt, allowing the system to consider both prompts and refine the search results accordingly.

To start a new NLP search, you simply do not use an nlpId in the request.

Extended Support for AI Image Search

If the user's prompt includes visual preferences, the NLP feature will generate a requestBody that incorporates AI image search functionality. This is particularly useful for refining searches based on aesthetic or design elements mentioned by the user.

Example with Visual Input:

Prompt:

Find me a condo in Toronto with at least 1 bedroom in the Annex. It must have underground parking and a white kitchen.

Request:

POST https://api.repliers.io/nlp


```
{
   "prompt": "Find me a condo in toronto with at least 1 bedroom in the annex. It must have underground parking and a white kitchen."
}
```


Response:


```
{
   "request": {
      "url": "https://api.repliers.io/listings?propertyType=Condo%20Apt&area=Toronto&minBeds=1&neighborhood=Annex&garage=Underground",
      "body": {
```

```
        "imageSearchItems": [
          {
            "type": "text",
            "value": "white kitchen",
            "boost": 1
          }
        ]
      },
      "summary": "Searching for a condo apartment in Toronto with at least 1 bedroom in the Annex. The property should have underground parking. Properties with white kitchens are preferred."
    },
    "nlpId": "12345"
}
```


The AI image search integrates the visual preference "white kitchen" into the search, enabling users to find properties that match their aesthetic desires.

AI-Powered Data Normalization

Our NLP feature also includes AI-powered data normalization, ensuring that user inputs match the acceptable values from the MLS®. This feature is crucial for maintaining consistency and accuracy when translating user prompts into API requests.

Example of Data Normalization:

Prompt:

Find me a condo with an enclosed balcony.

Request:

POST https://api.repliers.io/nlp

```
{
   "prompt": "Find me a condo with an enclosed balcony."
}
```

Response:

```
{
   "request": {
      "url": "https://api.repliers.io/listings?class=condo&balcony=encl",
      "body": null,
      "summary": "Searching for condos with enclosed balconies."
   },
   "nlpId": "12345"
}
```

In this case, "enclosed balcony" has been normalized to "encl," which is the expected MLS® value for the parameter balcony.

406 Not Supported Response

The /nlp endpoint converts natural language listing searches into structured requests for the Repliers API. If an irrelevant prompt is submitted (for example, "why is the sky blue"), the endpoint returns a 406 response.

Filtering NLP Sessions

You can filter NLP sessions using the GET /nlp endpoint. This endpoint allows you to retrieve all NLP prompts and the sessions they are associated with, identified by a unique nlpId for each session.

Available Filters:

clientId: Filter sessions based on the clientId to see all NLP sessions associated with a specific client.
nlpId: Filter by nlpId to retrieve details about a specific NLP session.

Pagination:

If you have a large number of NLP sessions, you can paginate through the results to efficiently manage and review the data.

For more detailed information on how to implement these filters and paginate results, please visit our API Reference.

How to Enable NLP Search

To enable NLP search functionality in your application, navigate to the API keys tab of the developer portal and select the API key you wish to configure. When editing the key settings, you'll find an option to enable NLP search capabilities. Note that this feature requires integration with OpenAI's services, so you'll need to provide a valid OpenAI API key during the setup process. Once configured, this will enable the NLP search feature.

Cost Considerations

Please note that using this feature will incur OpenAI API fees. The costs will vary depending on your usage volume. For detailed pricing information and to estimate your potential costs, please refer to OpenAI's API pricing page. We recommend monitoring your usage and setting up billing alerts to manage your OpenAI expenses effectively.

In Closing

The AI-powered NLP feature in our real estate APIs is designed to enhance user experience and improve search relevance. By converting natural language prompts into precise API requests, integrating visual preferences through AI image search, and normalizing data for MLS® compatibility, our NLP feature offers a powerful tool for real estate applications.

What questions does this article answer?

How do I turn natural language prompts (“find me a condo in Toronto…”) into Repliers API listing searches?
How does the NLP endpoint work and what does its response look like?
How can I preserve conversational context across multiple prompts using nlpId?
How can I use NLP together with AI image search (e.g., “white kitchen”) in the request body?
How does AI-powered data normalization map user wording to valid MLS® values?
What happens if I send prompts that are not about listing search?

Articles on: Property Search & Filtering
Filtering Listings Geo-Spatially Using the "map" Parameter
Our real estate API allows users to filter listings geo-spatially by using the "map" parameter. Users can pass in single polygon or multipolygon shapes to isolate listings within specified map boundaries. This feature is commonly used to display listings within a map's boundaries or to allow users to draw shapes on a map to filter listings.

Single Polygon Request

To filter listings within a single polygon, use the following request format:

Request:

```
POST https://api.repliers.io/listings
{
  "map": [    [      [        -79.3928178512883,        43.65790500294517       ],
      [        -79.40145579410431,        43.651653527879944       ],
      [        -79.38874165704074,        43.64991005052994       ],
      [        -79.3928178512883,        43.65790500294517       ]
    ]
```

```
  ]
}
```

Multipolygon Request

To filter listings within multiple polygons, use the following request format:

Request:

```
POST https://api.repliers.io/listings
{
  "map": [    [      [        -79.3928178512883,        43.65790500294517       ],
      [        -79.40145579410431,        43.651653527879944       ],
      [        -79.38874165704074,        43.64991005052994       ],
      [        -79.3928178512883,        43.65790500294517       ]
    ],
    [      [        -79.38538193864319,        43.65790207978401       ],
      [        -79.37818295748833,        43.656856957533705       ],
      [        -79.37794674628078,        43.663805792537715       ],
      [        -79.38538193864319,        43.65790207978401       ]
    ]
  ]
}
```

Using the mapOperator Parameter

By default, if a multipolygon is used, the API returns listings that fall within any of the polygons. However, you can control this behavior using the mapOperator parameter. Setting mapOperator to "AND" will return listings that fall within all the polygons.

Example Request with mapOperator:

```
POST https://api.repliers.io/listings?mapOperator=AND
{
```

```
  "map": [    [      [       -79.3928178512883,       43.65790500294517      ],
      [       -79.40145579410431,       43.651653527879944      ],
      [       -79.38874165704074,       43.64991005052994      ],
      [       -79.3928178512883,       43.65790500294517      ]
    ],
    [     [       -79.38538193864319,       43.65790207978401      ],
      [       -79.37818295748833,       43.656856957533705      ],
      [       -79.37794674628078,       43.663805792537715      ],
      [       -79.38538193864319,       43.65790207978401      ]
    ]
  ]
}
```

Combining map with Other Parameters

The map parameter and mapOperator parameter can be combined with other parameters to refine your search further. For example, you can filter listings by price range, property type, and location within specified map boundaries.

Example Request Combining map With Other Parameters:

```
POST https://api.repliers.io/listings?minBeds=4&city=New
York&maxPrice=1250000&minBaths=4
{
  "map": [    [      [       -79.3928178512883,       43.65790500294517      ],
      [       -79.40145579410431,       43.651653527879944      ],
      [       -79.38874165704074,       43.64991005052994      ],
      [       -79.3928178512883,       43.65790500294517      ]
    ],
    [     [       -79.38538193864319,       43.65790207978401      ],
      [       -79.37818295748833,       43.656856957533705      ],
      [       -79.37794674628078,       43.663805792537715      ],
      [       -79.38538193864319,       43.65790207978401      ]
    ]
  ]
}
```

Using GET Requests

The map parameter can also be passed in a GET request. However, due to URL length limitations, we recommend using POST requests for larger polygons or multipolygon shapes. For more information on this topic please refer to Handling Detailed GeoJSON Polygons In Repliers API Requests.

Example GET Request:

GET
https://api.repliers.io/listings?map=[[[-79.3928178512883,43.65790500294517],[-79.401455794
10431,43.651653527879944],[-79.38874165704074,43.64991005052994],[-79.3928178512883
,43.65790500294517]]]

Conclusion

Using the map parameter allows you to filter listings within specific geographic boundaries, enhancing your ability to target relevant properties. Whether you need to filter within a single polygon or multiple polygons, our API provides the flexibility to meet your needs. For more detailed shapes or combinations, we recommend using POST requests to avoid URL length limitations.

What questions does this article answer?

How do I filter listings within a polygon or multipolygon using the map parameter?
How do I send GeoJSON-like coordinates in the request body for map-based filtering?
What does the mapOperator parameter do and when should I use AND with multipolygons?
How can I combine map with other filters like price, beds, baths, and city?
When should I use POST vs GET for map-based filtering?

Articles on: Property Search & Filtering
Map Clustering Implementation Guide
Our real estate APIs feature a map clustering capability that optimizes the display of listings on a map. This feature significantly improves performance by clustering listings server-side, thereby reducing the need to handle large datasets on the front-end.

Requesting Clusters

To request clusters of listings, set the cluster parameter to true in your API request:

GET https://api.repliers.io/listings?cluster=true

Example Response Containing Clusters


```
{
   "aggregates": {
      "map": {
         "clusters": [
            {
               "count": 19585,
               "location": {
                  "latitude": 43.70534443195513,
                  "longitude": -79.36733720479269
               },
               "bounds": {
                  "bottom_right": {
                     "latitude": 43.633363377302885,
                     "longitude": -79.10189635120332
                  },
                  "top_left": {
                     "latitude": 43.83431699126959,
                     "longitude": -79.45311376824975
                  }
               },
               "map": [
                  [
                     [
                        -79.45311376824975,
                        43.83431699126959
                     ],
                     [
```

```
                -79.45311376824975,
                43.633363377302885
            ],
            [
                -79.10189635120332,
                43.633363377302885
            ],
            [
                -79.10189635120332,
                43.83431699126959
            ],
            [
                -79.45311376824975,
                43.83431699126959
            ]
        ]
    ]
},
{
    "count": 11050,
    "location": {
        "latitude": 43.683894202150654,
        "longitude": -79.59600708914154
    },
    "bounds": {
        "bottom_right": {
            "latitude": 43.58039586804807,
            "longitude": -79.45316883735359
        },
        "top_left": {
            "latitude": 43.83450788911432,
            "longitude": -79.8046560678631
        }
    },
    "map": [
        [
            [
                -79.8046560678631,
                43.83450788911432
            ],
            [
                -79.8046560678631,
                43.58039586804807
            ],
```

```
                    [
                      -79.45316883735359,
                      43.58039586804807
                    ],
                    [
                      -79.45316883735359,
                      43.83450788911432
                    ],
                    [
                      -79.8046560678631,
                      43.83450788911432
                    ]
                  ]
                ]
              }
            ]
          }
        }
}
```


Map-Only Experience

For a map-only experience, it's recommended to set listings to false to improve load times. This ensures that only cluster data is returned without the detailed listing data:

GET https://api.repliers.io/listings?cluster=true&listings=false

Cluster Precision

The clusterPrecision parameter controls the density of clusters:

Lower precision values result in fewer clusters with more listings per cluster.
Higher precision values result in more clusters with fewer listings per cluster.

The value ranges from 1 to 29. Adjust the clusterPrecision based on the map zoom level for optimal results. For example, many users set clusterPrecision to match the zoom value provided by mapping libraries like Google Maps or Mapbox.

Example Request

Here's an example that sets clusterPrecision to 10 and requests specific fields for clusters containing a single listing:

GET
https://api.repliers.io/listings?cluster=true&clusterFields=mlsNumber,listPrice,address.city&listings=false&clusterPrecision=10

Cluster Limit

The clusterLimit parameter allows you to limit the number of clusters returned in the response. This value can range from 1 to 200.

GET https://api.repliers.io/listings?cluster=true&clusterLimit=100

Single Listing Clusters & the clusterFields Parameter

When a cluster contains a single listing, the cluster includes additional fields for that listing. This is useful for displaying details like listPrice in map applications. You can specify which fields to return using the clusterFields parameter.

Example Request for Single Listing Clusters

GET
https://api.repliers.io/listings?cluster=true&clusterFields=mlsNumber,listPrice,address.city&listings=false&clusterPrecision=10

In this request we specify that for clusters containing a single listing we'd like to have the mlsNumber, listPrice and city values.

Example Cluster Containing A Single Listing & Listing Fields


```
{
   "count": 1,
   "location": {
      "latitude": 43.81639128550887,
      "longitude": -79.60495973937213
   },
   "bounds": {
      "bottom_right": {
         "latitude": 43.81639128550887,
         "longitude": -79.60495973937213
      },
      "top_left": {
         "latitude": 43.81639128550887,
         "longitude": -79.60495973937213
      }
   },
   "map": [
      [
         [
            -79.60495973937213,
            43.81639128550887
         ],
         [
            -79.60495973937213,
            43.81639128550887
         ],
         [
            -79.60495973937213,
            43.81639128550887
         ],
```

```
            [
               -79.60495973937213,
               43.81639128550887
            ],
            [
               -79.60495973937213,
               43.81639128550887
            ]
         ]
      ],
      "listing": {
         "mlsNumber": "N8372244",
         "listPrice": "4500.00",
         "address": {
            "city": "New York"
         }
      }
}
```


clusterListingsThreshold Parameter

The clusterListingsThreshold parameter controls the maximum cluster size for which detailed listing information is included in the API response. By default, detailed listing data is only included when a cluster contains exactly 1 listing (displayed in a listing object). However, setting clusterListingsThreshold to a higher value allows you to retrieve detailed information for clusters containing multiple listings, up to your specified threshold. For example, setting clusterListingsThreshold=5 will include a listings array with full property details for any cluster containing 5 or fewer listings. This optimization reduces the need for additional API requests when users interact with small clusters, improving performance and user experience. Clusters exceeding the threshold will only return the listing count without detailed property information.

Cluster Statistics

To include statistics in your clusters, set clusterStatistics to true and use the statistics parameter. This feature is ideal for creating heat maps that compare different geographies based on market data.

Example Request for Cluster Statistics

GET https://api.repliers.io/listings?cluster=true&statistics=avg-listPrice&clusterStatistics=true

In this request we specify that we'd like to see the average listPrice statistic for all listings within each cluster

Example Of Cluster Containing Statistics


```
{
   "count": 3,
   "location": {
      "latitude": 43.82331159431487,
      "longitude": -79.60820881649852
   },
   "bounds": {
      "bottom_right": {
         "latitude": 43.82331159431487,
         "longitude": -79.60820881649852
      },
      "top_left": {
         "latitude": 43.82331159431487,
         "longitude": -79.60820881649852
      }
   },
   "map": [
      [
         [
            -79.60820881649852,
            43.82331159431487
         ],
         [
            -79.60820881649852,
            43.82331159431487
         ],
         [
```

```
            -79.60820881649852,
            43.82331159431487
          ],
          [
            -79.60820881649852,
            43.82331159431487
          ],
          [
            -79.60820881649852,
            43.82331159431487
          ]
        ]
    ],
    "statistics": {
        "listPrice": {
            "avg": 300533
        }
    }
}
```


Conclusion

Our map clustering feature provides an efficient way to display large datasets on a map by aggregating listings server-side. Adjusting parameters like clusterPrecision, clusterFields, and clusterStatistics allows you to tailor the clustering to your specific needs, ensuring a smooth and responsive user experience.

What questions does this article answer?

How do I enable server-side map clustering with cluster=true on the /listings API?
What does the cluster response format look like (bounds, count, centroid, optional listing data)?
How can I use clusterListingsThreshold to control when detailed listing info is included in clusters?
How does clustering improve front-end performance for map views with many listings?
How do clusters relate to the aggregates.map.clusters structure?

Articles on: Property Search & Filtering
Map Clustering Implementation Guide

Our real estate APIs feature a map clustering capability that optimizes the display of listings on a map. This feature significantly improves performance by clustering listings server-side, thereby reducing the need to handle large datasets on the front-end.

Requesting Clusters

To request clusters of listings, set the cluster parameter to true in your API request:

GET https://api.repliers.io/listings?cluster=true

Example Response Containing Clusters


```
{
   "aggregates": {
      "map": {
         "clusters": [
            {
               "count": 19585,
               "location": {
                  "latitude": 43.70534443195513,
                  "longitude": -79.36733720479269
               },
               "bounds": {
                  "bottom_right": {
                     "latitude": 43.633363377302885,
                     "longitude": -79.10189635120332
                  },
                  "top_left": {
                     "latitude": 43.83431699126959,
                     "longitude": -79.45311376824975
                  }
               },
               "map": [
                  [
                     [
                        -79.45311376824975,
```

```
            43.83431699126959
          ],
          [
            -79.45311376824975,
            43.633363377302885
          ],
          [
            -79.10189635120332,
            43.633363377302885
          ],
          [
            -79.10189635120332,
            43.83431699126959
          ],
          [
            -79.45311376824975,
            43.83431699126959
          ]
        ]
      ]
    },
    {
      "count": 11050,
      "location": {
        "latitude": 43.683894202150654,
        "longitude": -79.59600708914154
      },
      "bounds": {
        "bottom_right": {
          "latitude": 43.58039586804807,
          "longitude": -79.45316883735359
        },
        "top_left": {
          "latitude": 43.83450788911432,
          "longitude": -79.8046560678631
        }
      },
      "map": [
        [
          [
            -79.8046560678631,
            43.83450788911432
          ],
          [
```

```
                    -79.8046560678631,
                    43.58039586804807
                ],
                [
                    -79.45316883735359,
                    43.58039586804807
                ],
                [
                    -79.45316883735359,
                    43.83450788911432
                ],
                [
                    -79.8046560678631,
                    43.83450788911432
                ]
            ]
        ]
    }
  ]
 }
}
```


Map-Only Experience

For a map-only experience, it's recommended to set listings to false to improve load times. This ensures that only cluster data is returned without the detailed listing data:

GET https://api.repliers.io/listings?cluster=true&listings=false

Cluster Precision

The clusterPrecision parameter controls the density of clusters:

Lower precision values result in fewer clusters with more listings per cluster.

Higher precision values result in more clusters with fewer listings per cluster.

The value ranges from 1 to 29. Adjust the clusterPrecision based on the map zoom level for optimal results. For example, many users set clusterPrecision to match the zoom value provided by mapping libraries like Google Maps or Mapbox.

Example Request

Here's an example that sets clusterPrecision to 10 and requests specific fields for clusters containing a single listing:

GET
https://api.repliers.io/listings?cluster=true&clusterFields=mlsNumber,listPrice,address.city&listings=false&clusterPrecision=10

Cluster Limit

The clusterLimit parameter allows you to limit the number of clusters returned in the response. This value can range from 1 to 200.

GET https://api.repliers.io/listings?cluster=true&clusterLimit=100

Single Listing Clusters & the clusterFields Parameter

When a cluster contains a single listing, the cluster includes additional fields for that listing. This is useful for displaying details like listPrice in map applications. You can specify which fields to return using the clusterFields parameter.

Example Request for Single Listing Clusters

GET
https://api.repliers.io/listings?cluster=true&clusterFields=mlsNumber,listPrice,address.city&listings=false&clusterPrecision=10

In this request we specify that for clusters containing a single listing we'd like to have the mlsNumber, listPrice and city values.

Example Cluster Containing A Single Listing & Listing Fields


```
{
   "count": 1,
   "location": {
      "latitude": 43.81639128550887,
      "longitude": -79.60495973937213
   },
   "bounds": {
      "bottom_right": {
         "latitude": 43.81639128550887,
         "longitude": -79.60495973937213
      },
      "top_left": {
         "latitude": 43.81639128550887,
         "longitude": -79.60495973937213
      }
   },
   "map": [
      [
         [
            -79.60495973937213,
            43.81639128550887
         ],
         [
            -79.60495973937213,
            43.81639128550887
         ],
         [
```

```
            -79.60495973937213,
            43.81639128550887
          ],
          [
            -79.60495973937213,
            43.81639128550887
          ],
          [
            -79.60495973937213,
            43.81639128550887
          ]
        ]
    ],
    "listing": {
        "mlsNumber": "N8372244",
        "listPrice": "4500.00",
        "address": {
            "city": "New York"
        }
    }
}
```


clusterListingsThreshold Parameter

The clusterListingsThreshold parameter controls the maximum cluster size for which detailed listing information is included in the API response. By default, detailed listing data is only included when a cluster contains exactly 1 listing (displayed in a listing object). However, setting clusterListingsThreshold to a higher value allows you to retrieve detailed information for clusters containing multiple listings, up to your specified threshold. For example, setting clusterListingsThreshold=5 will include a listings array with full property details for any cluster containing 5 or fewer listings. This optimization reduces the need for additional API requests when users interact with small clusters, improving performance and user experience. Clusters exceeding the threshold will only return the listing count without detailed property information.

Cluster Statistics

To include statistics in your clusters, set clusterStatistics to true and use the statistics parameter. This feature is ideal for creating heat maps that compare different geographies based on market data.

Example Request for Cluster Statistics

GET https://api.repliers.io/listings?cluster=true&statistics=avg-listPrice&clusterStatistics=true

In this request we specify that we'd like to see the average listPrice statistic for all listings within each cluster

Example Of Cluster Containing Statistics


```
{
   "count": 3,
   "location": {
      "latitude": 43.82331159431487,
      "longitude": -79.60820881649852
   },
   "bounds": {
      "bottom_right": {
         "latitude": 43.82331159431487,
         "longitude": -79.60820881649852
      },
      "top_left": {
         "latitude": 43.82331159431487,
         "longitude": -79.60820881649852
      }
   },
   "map": [
      [
         [
            -79.60820881649852,
            43.82331159431487
         ],
         [
```

```
            -79.60820881649852,
            43.82331159431487
          ],
          [
            -79.60820881649852,
            43.82331159431487
          ],
          [
            -79.60820881649852,
            43.82331159431487
          ],
          [
            -79.60820881649852,
            43.82331159431487
          ]
        ]
    ],
    "statistics": {
      "listPrice": {
        "avg": 300533
      }
    }
}
```


Conclusion

Our map clustering feature provides an efficient way to display large datasets on a map by aggregating listings server-side. Adjusting parameters like clusterPrecision, clusterFields, and clusterStatistics allows you to tailor the clustering to your specific needs, ensuring a smooth and responsive user experience.

What questions does this article answer?

How do I enable server-side map clustering with cluster=true on the /listings API?
What does the cluster response format look like (bounds, count, centroid, optional listing data)?
How can I use clusterListingsThreshold to control when detailed listing info is included in clusters?
How does clustering improve front-end performance for map views with many listings?
How do clusters relate to the aggregates.map.clusters structure?

Articles on: Property Search & Filtering

How to Search Listings Using Keywords

Our real estate APIs offer a robust keyword search feature that allows API users to search listings using specific keywords. This feature is highly versatile and can be used to create a general search input for your users, enabling them to find listings by MLS® number, address, or by searching text in descriptions. Additionally, you can implement an auto-complete function using this feature.

The "search" Parameter

The search parameter enables users to find listings that contain specific keywords anywhere in the listing details. For example, to find listings containing the keyword "cul-de-sac", you can make the following request:

```
GET https://api.repliers.io/listings?search=cul-de-sac
```

This request will return all listings where "cul-de-sac" appears in any part of the listing.

Search for Multiple Keywords or Phrases

You can search for multiple keywords or phrases by separating them with commas. This ensures that all specified terms are present in the listing data.

Example:

```
https://api.repliers.io/listings?search=cul-de-sac,high ceilings
```

This will retrieve listings where both "cul-de-sac" and "high ceilings" appear in the listing data.

Articles on: Property Search & Filtering

Searching Listings By Address

When using our API, you might need to search for property listings by a specific address. There are a couple of ways to do this. Below are the methods you can use to search for listings at a given address, such as 101 Yonge St.

Method 1: Use search and searchFields

This method allows you to search listings by specifying the address directly in the query. Here's how you can find listings at 101 Yonge St:

https://api.repliers.io/listings?search=101 Yonge St&searchFields=address.streetName,address.streetNumber

In this example:

search=101 Yonge St specifies the address you are searching for.
searchFields=address.streetName,address.streetNumber tells the API to search within the street name and street number fields.

Method 2: Use Address Parameters

Alternatively, you can use specific address parameters to search for listings. Here's how you can do this:

https://api.repliers.io/listings?streetName=Yonge&streetNumber=101

In this example:

streetName=Yonge specifies the street name.
streetNumber=101 specifies the street number.

Including Both Active and Unavailable Listings

By default, the above methods will only return active listings. If you want to search for both active and unavailable listings, you need to add a status parameter to your request. Here's how you can modify the second method to include both statuses:

https://api.repliers.io/listings?streetName=Yonge&streetNumber=101&status=["A","U"]

In this example:

status=["A","U"] includes both active (A) and unavailable (U) listings in the search results.

Summary

Method 1: Use the search parameter combined with searchFields.
Method 2: Use specific address parameters.
Include Both Active and Unavailable Listings: Add the status parameter with the values ["A","U"].

These methods provide flexibility depending on your needs and how you prefer to structure your API requests.

What questions does this article answer?

How can I search listings by a specific address using search + searchFields?
How can I search using explicit address parameters like streetName and streetNumber?
How do I include both active and unavailable listings by combining status and address filters?

When would I choose method 1 (keyword + fields) vs. method 2 (structured address parameters)?

The "searchFields" Parameter

The searchFields parameter allows you to isolate the search to specific fields. This is particularly useful for searches focused on addresses. For example, to find listings at "1 Yonge St", you can restrict the search to the address fields:

GET https://api.repliers.io/listings?search=1 Yonge St&searchFields=address.streetNumber,address.streetName

Exclude Specific Keywords or Phrases

To exclude listings containing specific keywords, use the not: prefix. This allows you to filter out unwanted terms from the results.

Example:

https://api.repliers.io/listings?search=cul-de-sac,high ceilings,not:bungalow&searchFields=details.description

This will retrieve listings where both "cul-de-sac" and "high ceilings" appear in the details.description field but exclude any listings where the word "bungalow" is present.

Summary of Search Parameters

search: Specifies the keywords or phrases to search for.
searchFields: Specifies which fields to search within (e.g., details.description).
not:: Excludes listings containing specific keywords or phrases.

The "fuzzySearch" Parameter

The fuzzySearch parameter, if set to true, accommodates small inconsistencies in user input. For example, if a user enters "236 Stratallan Wood" but they meant to type "236 Strathallan Wood", the search will still return relevant results for "Strathallan":

```
GET https://api.repliers.io/listings?search=236 Stratallan Wood&fuzzySearch=true
```

Combining Parameters

The search, searchFields, and fuzzySearch parameters can be combined with other search parameters to refine your query further. For example, to find listings at "1 Yonge St" with at least 3 bedrooms, you can use the following request:

```
GET https://api.repliers.io/listings?search=1 Yonge St&searchFields=address.streetNumber,address.streetName&minBeds=3
```

Conclusion

The keyword search feature in our real estate APIs provides powerful capabilities for searching listings by various criteria. By leveraging the search, searchFields, and fuzzySearch parameters, along with other search filters, you can offer a highly customizable search experience for your users. For further assistance or advanced implementation tips, please contact our support team.

What questions does this article answer?

How does the search parameter work for keyword-based listing search?
How do I search across descriptions, addresses, MLS® numbers, etc. using free text?
How can I limit keyword search to specific fields using searchFields?
How do I include or exclude specific words/phrases in a search?

What does fuzzySearch=true do and when should I use it?
Articles on: Property Search & Filtering
AI Image Search Implementation Guide
Our real estate APIs enable advanced image search capabilities using AI to enhance the search experience. When searching images, the results are not limited by the search criteria but are sorted by relevance, ensuring the most relevant listings appear first. Each listing is assigned a relevance score, indicating how closely it matches the image inputs.

Important Note

When using image search, the results are not restricted by the image search criteria. For instance, if you search through 500 listings, you will still receive all 500 listings in the response. The key benefit is that these listings will be sorted by relevance, with the most relevant listing appearing first. Each listing will have a score indicating its relevance to the image inputs, helping you quickly identify the best matches. By not limiting the results based on image inputs, we ensure that no potential listings are hidden from the buyer, saving them time and providing the most relevant options without excluding any inventory.

Using the API

To perform an image search, you will need to make a POST request to https://api.repliers.io/listings instead of a GET request. This is because the image inputs are provided in the body of the request in JSON format.

Request Structure

Text-based Image Search


```
{
   "imageSearchItems": [
      {
         "type": "text",
         "value": "wine cellar",
         "boost": 1
```

```
      }
   ]
}
```

Example: Searching for listings with a wine cellar.

Image-based Search

```
{
   "imageSearchItems": [
      {
         "type": "image",
         "url": "https://domain.com/swimmingpool-image-url.jpg",
         "boost": 1
      }
   ]
}
```

Example: Searching for listings with features similar to the provided swimming pool image.

Combining Multiple Inputs

You can provide multiple image inputs, both text-based and image-based, in a single request.

Multiple Text-based Inputs

```
{
   "imageSearchItems": [
      {
         "type": "text",
         "value": "wine cellar",
         "boost": 1
      },
```

```
      {
         "type": "text",
         "value": "white kitchen",
         "boost": 1
      },
      {
         "type": "text",
         "value": "sauna",
         "boost": 1
      }
   ]
}
```

Example: Searching for listings with a wine cellar, white kitchen, and sauna.

Combining Text-based and Image-based Inputs

```
{
   "imageSearchItems": [
      {
         "type": "text",
         "value": "wine cellar",
         "boost": 1
      },
      {
         "type": "text",
         "value": "white kitchen",
         "boost": 1
      },
      {
         "type": "image",
         "url": "https://domain.com/swimmingpool-image-url.jpg",
         "boost": 1
      }
   ]
}
```

Example: Searching for listings with a wine cellar, white kitchen, and features similar to the provided swimming pool image.

Boost Values

The boost value allows you to specify the importance of each input relative to the others. For instance, if a white kitchen is more important than the other features, you can assign it a higher boost value.

```
{
   "imageSearchItems": [
      {
         "type": "text",
         "value": "wine cellar",
         "boost": 1
      },
      {
         "type": "text",
         "value": "white kitchen",
         "boost": 10
      },
      {
         "type": "image",
         "url": "https://domain.com/swimmingpool-image-url.jpg",
         "boost": 1
      }
   ]
}
```

Example: Prioritizing white kitchen with a boost value of 10.

Combining with Other Criteria

Image inputs can be combined with other search criteria, such as property type, minimum bedrooms, and location. The image search will be performed only on listings that match the specified criteria.

Example Request with Additional Criteria

POST https://api.repliers.io/listings?city=New York&propertyType=Detached&minBeds=4


```
{
   "imageSearchItems": [
      {
         "type": "text",
         "value": "wine cellar",
         "boost": 1
      },
      {
         "type": "text",
         "value": "white kitchen",
         "boost": 10
      },
      {
         "type": "image",
         "url": "https://domain.com/swimmingpool-image-url.jpg",
         "boost": 1
      }
   ]
}
```


This request searches for detached properties with at least 4 bedrooms in New York, prioritizing listings with features similar to a white kitchen, wine cellar, and swimming pool.

Image Reordering

As an added benefit, we automatically reorder the images array so that the most relevant images based on the user's inputs are moved to the top. This ensures that the first images the user sees when scrolling through a listing are the most relevant, providing instant visual feedback to their image inputs.

AI Image Search Scoring Details

When using our AI Image Search feature, each listing in the response includes a score that indicates the listing's relevance concerning the input images provided (imageSearchItems).

Here’s a sample of how the score might appear in a response:


```
{...
   "updatedOn": "2024-07-10T11:52:52.000Z",
   "score": 0.949419,
   "images": [
      "IMG-C9031294_14.jpg"
   ]...
}
```


Understanding the Score:

Single Input Image:

For a search using one input image, the highest possible score (perfect match) is 1.0.

Multiple Input Images:

If you provide multiple images, the perfect score increases proportionally. For example:

Two input images: A perfect score is 2.0.
Three input images: A perfect score is 3.0.
And so on.

What the Score Represents:

The score is a numerical representation of how well a listing matches the provided input images. The closer the score is to the perfect score (1.0 per input image), the more relevant the listing is to the images you used in your search. This scoring system helps prioritize listings that best match your criteria, saving time and improving search relevance.

Example:

If you perform a search using three images, and a particular listing has a score of 2.85, this indicates a high degree of relevance to the input images, as the score is close to the perfect score of 3.0. This scoring mechanism is designed to enhance your experience by providing a clear, quantifiable measure of relevance.

Benefits of AI Image Search

Enhanced Search Relevancy: Saves users time by providing highly relevant search results.
Strong Personalization Feature: Allows for a tailored search experience based on user preferences.
Cutting-edge Technology: Utilizes advanced AI and vector technology for superior search performance.

Conclusion

Our AI Image Search feature enhances the search experience by providing highly relevant and personalized results. By leveraging advanced AI technology, users can save time and find the best listings that match their preferences.

What questions does this article answer?

How do I perform AI-powered image searches using the /listings endpoint with a POST body?
What is the structure of the imageSearchItems array and how do I send text vs. image inputs?
How are listings ranked by relevance score when using image search?

Can I combine multiple text and image inputs in a single image-search request?
Why does image search return all listings in scope but sort them by relevance instead of filtering them out?

Articles on: Property Search & Filtering
Sold Comparables & Similar (Active) Listings Selection Logic
Repliers makes it easy to get Sold Comparables and Similar (Active) Listings for any given property. This is particularly useful for creating a detailed property page, helping your users make more informed decisions with enriched information.

Sold Comparables

When a new listing hits the market, we check the MLS® history for similar listings. Here's what we consider when identifying comparables:

Status (sold, leased, or off-market)
Proximity to the new listing
Same number of bedrooms
Same transaction type (sale or lease)
Sold/leased within the past three months
Year built within +/- 10 years of the new listing
Square footage within +/- 200 sq. ft. of the new listing
Same property type or style
Sold/leased price within +/- 10% of the new listing's list price

Listings that match these criteria are included in the "comparables" section for the new listing. Here is an example of the comparables object:


```
{
   "comparables": [
      {
         "mlsNumber": "E11923411",
         "type": "Sale",
         "listPrice": 1159000,
```

```
        "soldPrice": 1450120,
        "soldDate": "2025-01-22T00:00:00.000-00:00",
        "address": {
          "streetNumber": "135",
          "streetName": "Woodycrest",
          "streetSuffix": "Avenue",
          "city": "Toronto",
          "neighborhood": "Danforth Village-East York",
          "zip": "M4J 3B7"
        },
        "distance": 0.4
      }
  ]
}
```


Working with Distance Data

Each comparable includes a distance field that shows how far the property is from the subject listing in kilometers. We return up to 20 comparables, prioritized by proximity among properties that meet all other criteria.

In some cases, comparables may be located further away than desired for your application. You can use the distance value to filter results based on your specific requirements—for example, only displaying comparables within a 5km radius.

Similar (Active) Listings

Our Get Similar Listings Endpoint finds active listings that are similar to a given property. Here's what we consider for these listings:

Status (active, on-market)
Same transaction type (sale or lease)
Same property type

Same number of bedrooms
Same neighborhood

You can also use optional parameters to refine your search:

radius: If specified, the search focuses on similar properties within this radius (in kilometers), ignoring the neighborhood criterion.
listPriceRange: If specified, the search includes properties with a list price within the given range.
For more details on optional parameters, please refer to our API Reference for Similar Listings.

Why Are Sold Comparables or Similar Listings Not Found?

If zero listings meet our selection criteria for Sold Comparables or Similar Listings, it's usually because the property in question is unique or located in a remote area. For instance, while it's common to find many Sold Comparables and Similar Listings for a downtown apartment, it may be challenging to find any for a luxury mansion in the countryside.

What questions does this article answer?

How does Repliers automatically select sold comparables for a listing?
What criteria are used for sold comparables (beds, year built, sqft, price range, timeframe, distance, property type)?
How should I interpret the comparables object and its distance field?
How are “Similar (Active) Listings” selected and what filters do they use?
How many comparables can I expect to receive and how are they ordered?