Geographic Address Management API

Table of Contents

Summary

The Geographic Address API provides a standardized client interface to an address management system. It allows searching for valid german addresses based on certain query parameters. These query parameters are parts of addresses like zip codes, city or street names. The response format can be controlled by special query parameters.

The API is based on the Telemanagement Forum Open Api TMF 673, V19.5 dated January 2020 with some adaptions for Deutsche Telekom.

The entrypoint for the API and the current API-documentation is available at: https://developer.telekom.de/catalog/geo/geo-geographicAddressManagement/g_api/production/3.0.0.

Business view - what is it for?

Many business processes require valid addresses, be it to deliver mail or devices or to provide a customer service at the right place. However, addresses are subject to constant change as new residential areas are added or old ones are changed. It is therefore important to always have access to an up-to-date stock, otherwise time-consuming manual activities will be necessary, for example, to deliver a postal item correctly.

Example of address search mask

Another example of such a process is the acquisition of new customers. A customer wants to know which products are actually being delivered to his address. In this process, a customer or call-center agent must be able to access all addresses known to the company as quickly as possible. By this the customer can precisly be offered which products and services are available at his address.

Addresses look very different...

Deutsche Telekom AG maintains a database with over 20 million address entries, which is permanently maintained by an editorial team and thus contains all valid geographical addresses in Germany. However, there are regional differences in the way addresses are composed. For example, districts are sometimes listed as part of the city name and sometimes separately. House number suffixes such as 'b', 'd-f' can be assigned like free text. Streets can be found as 'strasse', 'str', 'str.', etc.

The search for an exact address is therefore sometimes a bit difficult.

...and there are many ways to look for them.

When looking up addresses there is the possibility of simple typing errors when searching for an address e.g. look for a street in 'Franjfurt'.

Furthermore, spellings of streets and place names differ considerably from the spoken form. Thus, the 'Maurits-Ostyn-Weg' is spoken as 'Maurice-Oystin-Weg'. In telephone communication, e.g. a customer with a telephone hotline, this leads to several inquiries.

One API to catch it up

The API Geographic Address Management has implemented several methods to address these problems:

- **Timeliness**: up to four times a day, address changes of the address editor team are automatically added to the database
- **Near real-time response**: a query for an address is usually answered in less than 0.5s. This allows the API to be used in the implementation of interactive front-ends, such as an order portal or service portal.
- **Synonym search**: besides the exact spelling, synonym forms like 'str.', 'str', 'strasse', etc. are returned
- **Fuzzysearch**: also results of a similarity search will be returned by with the typing errors will be compensated. Thus the search for 'Franjfurt' returns all Frankfurts

By these possibilities a performant and flexible address search is possible with this API.

API features

Search for parts of addresses

If only parts of addresses are given to the query parameters the API will respond with a list of addresses best fitting to those values.

e.g. looking for a city by *Frank* will reveal *Frankfurt*, *Frankenheim*, *Frankenthal*, ...

Handling misspelling

The search mechanism of the API contains fuzzy logic to enable finding results even if the query parameter contains incorrect spelling.

e.g. looking for *Frangfurt* responds *Frankfurt am Main*, *Frankfurt (Oder)*

As there are nearly indefinite ways of misspelling this function has to balance between flexibility and amount of hits found. So if one experience unexpected responses keep in mind that the algorithm found a similarity in some way.

Controlling the response

The format of the API response can be controlled by special query fields. By these it is possible to limit the amount of data given back by the API.

e.g.

- *limit*: reduce number of datasets given back to this value
- *fields*: reduce the data fields in the response to these fields

Real-time processing

The API provides information requested in near real-time so that it can be used by consumers in an interactive way.

Usage Examples by functionality

Some usage examples will illustrated the functionality

Looking for all addresses of all streets in all cities with zip code starting with 28

```
GET https://serverRoot/geo/geographicAddressManagement/v3/geographicAddress?limit=100&postcode=28
```

Response

```
      "klsid": "14414152",
      "href": "http://serverRoot/geo/geographicAddressManagement/v3/geographicAddress/14414152",
      "city": "Bremen",
      "country": "DE",
      "locality": "Bremen",
      "postcode": "28195",
      "stateOrProvince": "BM",
      "streetName": "Abbentorstr.",
      "streetNr": "5",
      "geographicLocation": {
         "id": "KLS_14414152_53p08048_8p80058",
         "href": "n/a",
         "geometryType": "point",
         "geometry": [
           {
             "x": "53.08048",
             "y": " 8.080058",
             "z": "0"
           }
         ]
       },
      "onkz": "0421",
      "asb": "1"
    },...
```

Looking for all addresses in Bahnhofstr in Darmstadt

```
GET https://serverRoot/geo/geographicAddressManagement/v3/geographicAddress?
limit=100&postcode=64291&city=Darmstadt&street=Bahnhofstr
```

Response

```
    {
        "klsid": "9715573",
        "href": "http://serverRoot/geo/geographicAddressManagement/v3/geographicAddress/9715573",
        "city": "Darmstadt",
        "country": "DE",
        "locality": "Wixhausen",
        "postcode": "64291",
        "stateOrProvince": "HE",
        "streetName": "Bahnhofstr.",
        "streetNr": "1",
        "geographicLocation": {
          "id": "KLS_9715573_49p93279_8p64745",
          "href": "n/a",
        "geometryType": "point",
        "geometry": [
           {
             "x": "49.93279",
             "y": " 8.64745",
             "z": "0"
           }
        ]

        },
        "onkz": "06150",
        "asb": "6"
      },
      {...
```

Short list of zip codes in Berlin

```
GET https://serverRoot/geo/geographicAddressManagement/v3/geographicAddress?limit=100&fields=postcode&city=Berlin
```

Response

```
    {
        "postcode": "01936"
      },
      {
        "postcode": "09353"
      },
      {
        "postcode": "10115"
      },
      {
        "postcode": "10117"
      },
      {
        "postcode": "10119"
      },
      {
        "postcode": "10178"
      },
      {...
```

Search for zip codes and cities that sound like Bohn

```
GET https://serverRoot/geo/geographicAddressManagement/v3/geographicAddress?limit=100&fields=city%20postcode&city=Bohn
```

Response

```
        {
          "city": "Bonerath",
          "postcode": "54316"
        },
        {
          "city": "Bonese",
          "postcode": "29413"
        },
        {
          "city": "Bongard Eifel",
          "postcode": "53539"
        },
        {
          "city": "Bonn",
          "postcode": "53111"
        },
        {
          "city": "Bonn",
          "postcode": "53113"
        },
        {
          "city": "Bonn",
          "postcode": "53115"
        },...
```

Search for addresses by one string

```
GET https://serverRoot/geo/geographicAddressManagement/v3/geographicAddress?.fullText=Bahnhof Darmst
```

Response

```
        {
          "city": "Darmstadt",
          "streetName": "Bahnhofstr."
        },
        {
           "city": "Darmstadt",
           "streetName": "Am Alten Bahnhof"
        },
        {
          "city": "Roßdorf b Darmstadt",
          "streetName": "Bahnhofstr."
        },
        {
          "city": "Roßdorf b Darmstadt",
          "streetName": "Alte Bahnhofstr."
        },...
```

Endpoints

All URIs are relative to *https://serverRoot/geo/geographicAddressManagement/v3* serverroot is *stargate.prod.dhei.telekom.de*

Class	Method	HTTP request	Description
GeographicAddressApi	listGeographicAddress	**GET** /geographicAddress	List or find GeographicAddress objects
GeographicAddressApi	retrieveGeographicAddress	**GET** /geographicAddress/{klsid}	Retrieves a GeographicAddress by ID

listGeographicAddress

This is the master method for retrieving valid geographical addresses.

Parameters

Name	Type	Description	Notes
offset	**Integer**	Requested index for start of resources to be provided in response	[optional] [default to 0]
limit	**Integer**	Requested number of resources to be provided in response	[optional] [default to 100]
fields	**String**	List of fields limiting the response to this fields.	[optional] [default to null]
postcode	**String**	Postcode or part of postcode for search query	[optional] [default to null]
city	**String**	City name or part of name for search query	[optional] [default to null]
street	**String**	Street name or part of name for search query	[optional] [default to null]
streetNr	**String**	Street number for search query	[optional] [default to null]
authorization	**String**	Access token	[optional] [default to null]

Return type

List

Authorization

OAuth2 based as describes in master document.

HTTP request headers

- **Content-Type**: Not defined
- **Produces**: application/json

retrieveGeographicAddress

This operation retrieves a GeographicAddress entity given by klsid. All values according to the resource model are returned.

Parameters

Name	Type	Description	Notes
klsid	**String**	Identifier of the GeographicAddress	[default to null]
authorization	**String**	Access token	[optional] [default to null]

Return type

GeographicAddress

Authorization

AOAuth2 based as describes in master document.

HTTP request headers

- **Content-Type**: Not defined
- **Produces**: application/json

Parameters for searching addresses

Search values can be passed by query parameters to find matching valid addresses. The response format is controlled by query fields. Query parameter names are case-sensitive.

postcode

A zip-code or a left part of it can be given. Starting from the first number given as query parameter the service will give back a list of address entries fitting to that zip-codes. The match will be done starting from the left of postcode.

Example:

```
GET https://serverRoot/geo/geographicAddressManagement/v3/geographicAddress?postcode=28
```

retrieves a list of postcodes starting with "28" (e.g. 28192, 28217, ...) but not 32805.

city

A city name or a part of it can be given. Starting from the first character given as query parameter the service will give back a list of address entries fitting the city query. The query parameter will be searched in any new part of possible city names separated by space starting leftmost. The responded list will be ordered by a scorevalue resulting from elastic search query. The search in case insensitive.

Example:

```
GET https://serverRoot/geo/geographicAddressManagement/v3/geographicAddress?city=stadt
```

retrieves eg. Stadtallendorf, Stadtlohn, ..., Karlstadt, Neustadt, ...

City names in Germany are not created the same way throughout the country. There are a lot of local differences regarding parts a city name consists of. Some local communities add names of town districs directly to the city name other put it in seperate fields. In order to simplify the use of this API the term given in the *city* query parameter will be searched through all available city-related address fields, i.e. city names and district names. The returned dataset for a special address will show this parts separated.

Example (*fields* and *.fuzzy* see Parameters for formatting the response)

```
GET https://serverRoot/geo/geographicAddressManagement/v3/geographicAddress/?city=Bonn&fields=city&.fuzzy=false
```

retreives e.g. Bonn, Bonn Beuel, Bonn Bad Godesberg, ...

But the complete dataset of an address in Bonn Beuel shows this information in separate fields

```
GET https://serverRoot/geo/geographicAddressManagement/v3/geographicAddress/?city=Bonn Beuel&streetName=Agnesstr.&streetNr=1
```

retreives

```
{
        "klsid": "16827911",
        "href": "/geographicAddressManagement/v2/geographicAddress/16827911",
        "city": "Bonn",
        "country": "DE",
        "locality": "Beuel",
        "postcode": "53225",
        "stateOrProvince": "NW",
        "streetName": "Agnesstr.",
        "streetNr": "12",
        "geographicLocation": {
            "id": "KLS_16827911_50p73719_7p11573",
            "href": "n/a",
            "name": "n/a",
            "geometryType": "point",
            "geometry": {
                "x": "50.73719",
                "y": "7.11573",
                "z": "n/a"
            }
        },
        "onkz": "0228",
        "asb": "46"
    }
```

streetName

The response will contain all streetnames fitting to the given characters in the street-query parameter on the leftmost characters. The response will be given from the first character on. The query parameter will be searched in any new part of possible street names separated by space starting leftmost. The responded list will be ordered by a scorevalue resulting from elastic search query.
The search in case insensitive.

Example:

```
GET https://serverRoot/geo/geographicAddressManagement/v3/geographicAddress?postcode=28195&city=Bremen&streetName=Bahnhof
```

retrieves all addresses in Bahhofstraße in Bremen

streetNr

The response will give back all possible values of numbers. As like with the *city* parameter for ease of API usage the search will be done over a combination of street number and street number supplement. e.g. 12a, 3-7, 23g The full response body of a unique address will contain this information in separate fields

.fullText

A free string containing parts of an address can be entered here and a fulltext search will be performed over the entire database. When used the parameters *fields*, *postcode*, *city*, *streetName* and *streetNr* will be ignored. The results will be given back from the first letter but restricted to the number of datasets given in the limit-parameter. Results can be aggregated by *fields* parameter.

Parameters for formatting the response

limit

Limits the number of records contained in the API response it can range from 1 to 100. If no limit is given, the default of 10 is assumed. For a search without using the *fields*-parameter this limit simply applies to the list ordered by elastic search scores from the beginning on. Together with *offset*-parameter a pagination of results is possible. If *limits* is used together with *fields* the limit will only effect the number of results for the first to aggregation levels. The other levels of aggregation will be computed to fit to the maximum possible value for the search algorithm.

Example without using *fields*:

```
GET https://serverRoot/geo/geographicAddressManagement/v3/geographicAddress?city=Bremen&limit=100
```

retrieves the first 55 addresses in Bremen

Example using *limits* together with *fields*:

```
GET https://serverRoot/geo/geographicAddressManagement/v3/geographicAddress?city=Bremen&limit=100&fields=postcode streetName
streetNr
```

retrieves the first 100 postcodes in Bremen, for each postcode 30 street names and for each street name 30 street numbers

offset

offset can be used for pagination if the *fields*-parameter is not used. It sets the starting point for returned records within the complete list of results. example:

```
GET https://serverRoot/geo/geographicAddressManagement/v3/geographicAddress?city=Bremen&limit=25&offset=300
```

gives addreses 300 to 324 within Bremen. In conjunction with *fields* this parameter does not have an effect.

fields

Limits the amount of data given in the response. It is possible to request up to four of the allowed values (postcode, city, locality, streetName, streetNr, streetNrSuffix) to reduce the amount of response data. The returned list will only contain the fields given here with distinct operator applied. This means that each combination of requested values occurs only once. The list will be sorted according to elastic search score. A field value must not be given multiple times. Field values are case-sensitive.

examples: `postcode=283&fields=city postcode` returns a list of postcodes and city names of all postcodes starting with 283.

`postcode=283&fields=postcode city` returns an ascending list of postcodes and city names of all postcodes starting with 283.

As each field value leads to a new aggregation level (e.g. `fields=postcode city street` are 3 aggregation levels) that must be computed. By this the internal processing and search time rises fast with that number. It is recommended to

- use only little number of field values (one or two)
- reduce the limit value as much as possible
- use query values of at least 3 characters

.fuzzy (deprecated)

Allows to switch off fuzzy search. The default is true, i.e. a similarity search (aka spelling correction) will be done. When set to false a simple pattern matching search will be done. .fuzzy is deprecated and shall be substituted by .searchtype.

.searchtype

Determines the kind of search that is applied and can be one of *fuzzy*, *fragment*, *exact*

Only one of these are allowed per API call. Values are case-sensitive. If no *.searchtype* Parameter is given, *fuzzy* will be taken as default.

This parameter overrules the deprecated *.fuzzy* parameter. If both parameters are given in one search *.fuzzy* will be ignored.

fuzzy performs a full fuzzy featured search. Example using *fuzzy*:

```
GET https://serverRoot/geo/geographicAddressManagement/v3/geographicAddress?city=Berl&.searchtype=fuzzy
```

retrieves "Berlin", "Bernau", "Berglen"

fragment does search for fragments that a word starts with without applying any fuzzy logic. Example using *fragment*:

```
GET https://serverRoot/geo/geographicAddressManagement/v3/geographicAddress?city=Berl&.searchtype=fragment
```

retrieves "Berlin", "Berleburg" but not "Bernau". This is the same behaviour as *.fuzzy=false*.

exact retrieves exact matches only. Example using *exact*:

```
GET https://serverRoot/geo/geographicAddressManagement/v3/geographicAddress?city=Berl&.searchtype=exact
```

results in an empty response as there is no city *Berl*

Resource Models

- Error
- GeographicAddress
- GeographicLocationRefOrValue
- Point

Error Properties

Name	Type	Description	Notes
type	URI	A URI reference [RFC3986] that identifies the problem type. Consumers MUST use the *type* string as the primary identifier for the problem type.	[default to about:blank]
title	String	A short, human-readable summary of the problem type. Consumers SHOULD NOT automatically dereference the type URI.	[default to null]
status	Integer	The HTTP status code ([RFC7231], Section 6) generated by the origin server for this occurrence of the problem. The *status* member, if present, is only advisory; it conveys the HTTP status code used for the convenience of the consumer. Generators MUST use the same status code in the actual HTTP response.	[optional] [default to null]
detail	String	A human-readable explanation specific to this occurrence of the problem. The *detail* member, if present, ought to focus on helping the client correct the problem, rather than giving debugging information.	[optional] [default to null]
instance	URI	A URI reference that identifies the specific occurrence of the problem. It may or may not yield further information if dereferenced.	[optional] [default to null]

GeographicAddress Properties

Name	Type	Description	Notes
klsid	**String**	Unique identifier of the place This is Identical to the klsid at Telekom.	[optional] [default to null]
href	**String**	Unique reference of the place	[optional] [default to null]
city	**String**	City that the address is in	[optional] [default to null]
country	**String**	Country that the address is in	[optional] [default to null]
locality	**String**	An area of defined or undefined boundaries within a local authority or other legislatively defined area, usually rural or semi rural in nature. [ANZLIC-STREET], or a suburb, a bounded locality within a city, town or shire principally of urban character [ANZLICSTREET]	[optional] [default to null]
name	**String**	A user-friendly name for the place, such as [Paris Store], [London Store], [Main Home]	[optional] [default to null]
postcode	**String**	descriptor for a postal delivery area, used to speed and simplify the delivery of mail (also know as zipcode)	[optional] [default to null]
stateOrProvince	**String**	the State or Province that the address is in	[optional] [default to null]
streetName	**String**	Name of the street or other street type	[optional] [default to null]
streetNr	**String**	Number identifying a specific property on a public street. It may be combined with streetNrLast for ranged addresses	[optional] [default to null]
streetNrLast	**String**	Last number in a range of street numbers allocated to a property	[optional] [default to null]
streetNrLastSuffix	**String**	Last street number suffix for a ranged address	[optional] [default to null]
streetNrSuffix	**String**	the first street number suffix	[optional] [default to null]
streetSuffix	**String**	A modifier denoting a relative direction	[optional] [default to null]

Name	Type	Description	Notes
streetType	String	alley, avenue, boulevard, brae, crescent, drive, highway, lane, terrace, parade, place, tarn, way, wharf	[optional] [default to null]
geographicLocation	GeographicLocationRefOrValue		[optional] [default to null]
onkz	String	Dialing area-code	[optional] [default to null]
asb	String	Area connection ID (Anschlussbereich)	[optional] [default to null]

GeographicLocationRefOrValue Properties

Name	Type	Description	Notes
id	String	Unique identifier of the place	[optional] [default to null]
href	String	Unique reference of the place	[optional] [default to null]
name	String	A user-friendly name for the place, such as [Paris Store], [London Store], [Main Home]	[optional] [default to null]
geometryType	String	A string. Type allows describing Geographic Location form.	[optional] [default to null]
geometry	Point		[optional] [default to null]

Point Properties

Name	Type	Description	Notes
x	String	x coordinate (latitude)	[optional] [default to null]
y	String	y coordinate (longitude)	[optional] [default to null]
z	String	z coordinate (elevation)	[optional] [default to null]

Authorization

Authorization is OAuth2 based. Each consumer must first be registered by the API team and by this gets its secret-keys. In order to do so use contact information

Access tokens are provided by this URL:

https://iris.prod.dhei.telekom.de/auth/realms/default/protocol/openid-connect/token

The token has to be sent in request header as usual.

Contact

The API team of Geographic Address Management can me contacted by mail

Geo-Hub_Borecole@internal.telekom.com