Print-Mailing Targeting API

v 1.1.0-168-43286a6
Division: Post & Parcel Germany, Post

This API is used for planning **Dialogue Marketing** print mailings.

- Plan new **target groups** and
- Select **target areas**.
- Get possible options for your print mailing directly and
- Receive relevant **recipient data** for the dispatch preparation.

Region: Germany
Used for: Shipping, Advertising

Overview

With the **Print-Mailing Targeting API**, you plan the target group for your print mailing. Plan new target groups, select target regions and **directly receive the potential number of recipients** for shipping with the Print-Mailing APIs:

- Issuance of target region data, e.g. by postal code or town
- Issuance of target group data by relevant criteria, e.g. purchasing power
- Generation of the selection data for further processing, e.g. for dispatch preparation

The web services covers the following postal products of DPAG:

- POSTAKTUELL.
- POSTWURFSPEZIAL

The **benefits** for you:

- Plan the target groups for your print mailshot with one software for all Deutsche Post dialogue marketing products
- Full integration into your system landscape and process flows
- No license fees
- New functions, data and prices without time-consuming software updates
- Compliant with GDPR and Postal Act
- Data processing in Germany and the EU (private cloud of Deutsche Post DHL)

For more information, see https://www.deutschepost.de/de/p/print-mailing/partner.html.

Audience information

The primary audience of this documentation are readers with a technical background, who are interested in

- estimating the complexity of integrating the service
- understanding the business and technical requirements and
- integrating the respective APIs to the service.

Get Access

The most API resources are freely available to you, without having credentials:

- Retrieving Data for your selection
- Using temporary target-group selections

In order to **process your target-group selection** further to dispatch-preparation or booking (see this chapter), you need to have an own user environment and credentials.

- for **Automation**, please consult the Print-Mailing Automation API
- for **Dispatch preparation**, please consult the Print-Mailing Dispatch Preparation API

Environments

We offer two environments which may differ in their version and documentation

- Production environment (PROD)
    - use this Documentation
- Test Environment (UAT)
    - use this Documentation for UAT
    - serves as a sandbox
    - *Requests made to the test environment never affect productive systems and are not billed*.

See the servers drop-down menu in the Reference Docs for the environments URLs.

There are two different technical specifications for both the production environment (PROD) and test environment (UAT). Updates will be available in UAT first. These updates then also appear with a time lag on PROD. You can see from the release notes of both documentations whether the technical specification for the UAT is more up-to-date than for the PROD.

Using the API

This API is built conform to the REST architectural style to provide interoperability.

Exposed API resources can be manipulated using HTTP methods. Requests to a resource URL generate a response with JSON payload containing detailed information about the requested resource. HTTP response codes are used to indicate API errors.

We use a token-based approach to authorize requests coming from your system.

Http requests

Apply the standardized HTTP method semantics to communicate with the Print-Mailing Automation API.

The API offers clearly identifiable **resources** ("things of the Print-Mailing Automation universe"). You can use a resource's **representation** to create, update, delete and execute resources. There are different types of resources:

- **Single resource**: E.g. a systemuser, a campaign and its corresponding properties. A PUT updates properties of an existing single resource identified by an id.
- **List resource**: A (pageable) list of single resources e.g. a list of systemusers. Usually, a list resource doesn't include all properties of the single resources. A POST adds a new single resource to a list resource.
- **Processing resource**: Triggers a usually long running asynchronous process. This might be a price calculation or a campaign order. A POST is used to trigger processing resources.
- **Lookup resource**: Offer permitted (enum) field values for simpler communication with the API. API clients can see which values they can expect and have to include in requests. E.g. values for campaign states (active, paused, ...). Usually, lookup resources include an internationalized label.

POST requests are used to create single resources on a list resource. The semantic for list endpoints is best described as 'please add the enclosed representation to the list resource identified by the URL'. On a successful POST request, the server will create one or multiple new resources and provide their URLs in the response. Successful POST requests will usually generate 201.

PUT requests are used to update an **entire** resources, which already exists and thus can be clearly identified by an id. The semantic is best described as "please put the enclosed representation at the resource mentioned by the URL, replacing any existing resource". On successful PUT requests, the server will replace the entire resource addressed by the URL with the representation passed in the payload (subsequent reads will deliver the same payload). Successful PUT requests will usually generate 200.

GET, **DELETE** and **OPTIONS** requests follow the standardized HTTP method semantics.

Headers

You can (and sometimes have to) set HTTP headers. This could be the 'Content-Type', 'Accept', 'Accept-Encoding' or 'Authorization' headers, among others. Please only set headers that are necessary.

Caution

Be aware that HTTP headers with empty values are not allowed for security reasons and such requests will be rejected.

Payload and media types

The standard media type for requests and responses is **application/json** with default encoding **UTF-8**. However, other media types like **application/xml** are supported, too. See the concrete resource definition for all accepted media types.

Tip

Set the header **'Accept: application/json'** in your JSON requests in order to get a detailed JSON response in case of an error.

Use the same semantics for **null** and **absent** properties. OpenApi 3.x allows to mark properties as required and as nullable to specify whether properties may be absent ({}) or null ({"example":null}). If a property is defined to be not required and nullable (see 2nd row in Table below), this rule demands that both cases must be handled in the exact same manner by specification.

The following table shows all combinations and whether the examples are valid:

required	**nullable**	**{}**	**{"example":null}**
true	true	No	Yes
false	true	Yes	Yes
true	false	No	No

false	false	Yes	No

boolean properties must not be null. A boolean is essentially a closed enumeration of two values, true and false.

HTTP response compression

The Print-Mailing Automation API will compress all text-based responses using your favorite compression algorithm.

Tip

Please set a correct **Accept-Encoding** http header for all your request, if applicable!

This increases the API user experience and saves bandwidth.

Date and time formats

Use the date and time formats defined by RFC 3339:

- For "date" use strings matching yyyy-MM-dd, for example: 2026-05-28
- For "date-time" use strings matching yyyy-MM-dd'T'HH:mm:ss.SSS'Z', for example 2026-05-28T14:07:17.000Z

Note that the OpenApi format "date-time" corresponds to "date-time" in the RFC and "date" corresponds to "full-date" in RFC. Both are specific profiles, a subset of the international standard ISO 8601.

A zone offset must not be used. Date time values are always interpreted in **UTC** without offsets. Localization of date time should be done by the api client, if necessary.

Status codes and common error structures

Apply the standardized HTTP status code semantics to communicate with the Print-Mailing Automation API.

The OpenApi specification of every resource provides information about specific success and error responses. It is not a complete list to avoid an overload with common sense information. See below the list of most commonly used status codes in the Print-Mailing Automation API:

Status code	Meaning
200 OK	Request completed successfully.
201 Created	Request completed successfully. Resource created.
400 Bad Request	Invalid/not parseable JSON payload. Non existing enum value recognized. Concurrent update of a resource.
401 Unauthorized	Missing access token in the request headers.
403 Forbidden	Access denied because of missing authority. E.g. customer id is not allowed.
404 Not Found	The resource is not found.
422 Unprocessable Entity	The request could be parsed but the contents are invalid. See the business validation error code for further information.

500 Internal Server Error	A generic error indication for an unexpected server execution problem.
503 Service Unavailable	Service is (temporarily) not available.

In case of an error or input data validation failure, the services will respond with the following error representation in the response body (including internationalized messages).

For general errors (e.g. service unavailable):


```
{
  "timestamp": "2026-02-18T16:11:13.456Z",
  "status": 503,
  "correlationId": "3ea93639-3d7f-4a5c-88ee-f6b6d01dffac",
  "error": "Service Unavailable",
  "errors": [{
    "errorCode": "DM_CO_0012",
    "errorMessage": "The requested functionality is currently not available.Try again later.",
    "arguments": [],
    "fieldName": null
  }]
}
```


For specific validation of a sent field:


```
{
  "timestamp": "2026-02-18T16:12:05.352Z",
  "status": 422,
  "correlationId": "08a884be-3aec-4fa5-b647-7f82149144d5",
  "error": "Unprocessable Entity",
  "errors": [{
    "errorCode": "NotBlank",
    "errorMessage": "must not be empty",
    "arguments": [],
```

```
    "fieldName": "password"
    }]
}
```

Tip

Notice the unique **errorCode**: It describes a specific error case which is documented in the OpenApi. Let your application handle these specific error cases and don't only look for http status codes!

If you want to get in contact with us because of an error response, please also supply the **correlationId**. This is a unique id which is generated for every request and can help us to trace requests through our backend.

Common representation structures

Most of the resource representations include common fields:

```
{
   "id":87075,
   "createdOn":"2020-04-02T11:48:51.000Z",
   "changedOn":"2020-04-29T13:02:56.000Z",
   "version":44,
   ".....": "....."
}
```

Use this **id** to address the resource. The **version** is a simple counter informing about resource changes.

A **paged list resource** has the following structure:

```
{
   "elements":[
      {
         "id":549,
         "createdOn":"2020-03-04T09:05:11.000Z",
         "changedOn":null,
         "version":1,
         ".....": "....."
```

```
        },
        { ... }
    ],
    "page":{
        "size":50,
        "totalElements":13,
        "totalPages":1,
        "number":0
    },
    "links":[
        {
            "rel":"self",

"href":"https://print-mailing-api.deutschepost.de/user/customers
?page=0&size=50&sort=company,asc"
        }
    ]
}
```

Pagination and filtering list resources

Several list resources of the Print-Mailing Automation API support pagination and filtering to reduce the amount of returned data and improve the API client experience. If no resource fits to the given filtering criteria an empty list is returned. An HTTP 404 is never returned form a list resource.

Request	Meaning
`?page=0&size=50`	Get the first page, maximum 50 items per page.

`?page=0&size=50&company=Dynp`	Get the first page, maximum 50 items per page. Search in field "company" with value "Dynp".
`?page=0&size=50&sort=company,asc`	Get the first page, maximum 50 elements per page. Sort ascending by fieldName "company".
`?sort=company,asc&sort=type,desc`	Get the default number of elements per page. Sort ascending by fieldName "company" and descending by "type".

Caution

The maximum number of returned elements per list resource is always **2000**. You MUST use paging, if you expect bigger result sets.

List resources also include **link relation fields** for easier pagination:

- **self**: The hyperlink pointing to the same page.
- **prev**: The hyperlink pointing to the previous page.
- **next**: The hyperlink pointing to the next page.

Lookup resources

Lookup resources offer permitted (enum) field values for simpler communication with the API. There are cases where a API client has to send a specific value from a set of permitted values. If you have a frontend application, these kind of values usually can be picked from a drop down menu.

A lookup resource publishes all of this permitted values (which are mostly ids) together with an internationalized label.

```
{
   "Font": [{
      "id": 10,
      "label": "Arial"
   },
   {
      "id": 11,
      "label": "Arial Bold"
   }, "...."]
}
```

Error code lookup resources

Error code lookup resources offer a complete list of error codes, specific resources may return in an error response. This resource can be helpful during API client development. An error code lookup resource publishes all of this codes with an internationalized label.

```
{
   "CommonBusinessError":[
      {
         "value":"DM_CO_0001",
         "label":"Auswahlwert existiert nicht."
      }, "..."
   ],
   "RecipientBusinessError":[
      {
         "value":"DM_RC_0001",
         "label":"Die Adressfelder des Print-Mailings sind nicht
definiert."
      },
      {
         "value":"DM_RC_0002",
         "label":"Das Print-Mailing existiert nicht."
      }, "..."
   ]
}
```

Error code lookup resources may include **common business errors**. This type of errors is common to all resources.

Versioning

The API has explicit API versioning within the URL (`/v1/`). However, we apply the following rules to evolve the Print-Mailing Automation API in a backward-compatible way to stay on v1 as long as possible:

- Add only optional, never mandatory fields.
- Never change the semantic of fields (e.g. changing the semantic from customer-ekp to customer-id, as both are different unique customer keys)
- Input fields may have (complex) constraints being validated via server-side business logic. Never change the validation logic to be more restrictive and make sure that all constraints are clearly defined in description.
- Support redirection in case an URL has to change 301 (Moved Permanently).

API clients should apply the following principles:

- Be conservative with API requests and data passed as input, e.g. avoid to exploit definition deficits like passing megabytes of strings with unspecified maximum length.
- Be tolerant in processing and reading data of API responses.
- Be tolerant with unknown fields in the payload, i.e. ignore new fields **but do not eliminate them from payload if needed for subsequent PUT requests**. Remember, the semantics of PUT.
- Be prepared that enum return parameter may deliver new values; either be agnostic or provide default behavior for unknown values.
- Be prepared to handle HTTP status codes not explicitly specified in endpoint definitions. Note also, that status codes are extensible. Default handling is how you would treat the corresponding 2xx code (see RFC 7231 Section 6).
- Follow the redirect when the server returns HTTP status code 301 (Moved Permanently).

API resources and workflow steps

This section walks you through the resources currently available.

Retrieving Data for your selection

Via "GET" Endpoints

The Targeting API provides multiple ways of retrieving the data you need for your selection via a variety of requests using the HTTP "GET" method.

For the following example you use the endpoint:

/postal-code/{postal code}

where **{postal code}** is replaced by postal code of your choosing.

```
GET
https://print-mailing-api.deutschepost.de/targeting/postal-code/
52349

HTTP/1.1 200
{
  [
    {
        "id": "52349",
        "type": "Feature",
        "properties": {
            "id": "52349",
            "centroid": {
                "type": "Point",
                "coordinates": [
                    6.465422646782157,
                    50.80030381672309
                ]
            },
            "hh": 7167,
            "postal_code": "52349"
        },
        "geometry": {
            "type": "MultiPolygon",
            "coordinates": [
                [
                    [
                        [
```

```
                                6.479564800593006,
                                50.80942448326762
                            ],
                            .
                            .
                            .multiple coordinates are following
                            .
                            .
                            [
                                6.479564800593006,
                                50.80942448326762
                            ]
                        ]
                    ]
                ]
            }
        }
]
}
```

All other data retrieving "GET" endpoints work in the same way as the depicted example. They all return data about the geometric shape and meta information regarding the provided parameter.

Below, you find a list of the available endpoints and information about the data they return.

Endpoint	Parameter	Data returned
/zone	None	Returns geometric information on all zones. A zone comprises all households with the same first digit of the postal code.

/zone/{zone}	Zone ID	Returns information for a specific zone.
/region/{region}	Region ID	Returns information on a specific region shape. A region comprises all households with the same first two digits of the postal code.
/postal-code/{postal-code}	Postal code	Returns information for a specific postal code
/neighbourhood/{neighbourhoodId}	Neighbourhood ID	Returns information for a specific neighbourhoodId.
/delivery-district/{delivery-district}	Delivery district	Returns information for a specific delivery district

Via "POST" Endpoints

Compared to the "GET" requests, the HTTP "POST" endpoints differ only in the way information is fed to them.

Here, you can search by using a number of parameters instead of only one parameter specific to the request.

In the following example, you want to receive delivery-district shapes. Therefore, you use the endpoint:

/delivery-district/search

For the body, you provide different parameters that you want data on:

```
{
  "cities": [
    {
      "city": "Hamburg",
      "district": "Sternschanze",
      "latitude": 6.2345,
      "longitude": 50.6789
    }
  ],
  "neighbourhoods": [
    "51706"
  ],
  "districtCodes": [
    "52222-21"
  ],
  "postalCodes": [
    "48231"
  ],
  "regions": [
    "64"
  ],
  "zones": [
    "6"
  ],
  "coordinates": [
    {
      "latitude": 6.2345,
      "longitude": 50.6789,
      "distance": 5
    }
  ],
  "ags": [
    "18807060"
  ]
}
```

Note that you can mix and match your search parameters as you please. you can even use delivery-districts themselves as search parameters.

Here is the full example with request and response:

```
POST
https://print-mailing-api.deutschepost.de/targeting/delivery-dis
trict/search
Content-Type: application/json

{
  "cities": [
    {
      "city": "Hamburg",
      "district": "Sternschanze",
      "latitude": 6.2345,
      "longitude": 50.6789
    }
  ],
  "neighbourhoods": [
    "51706"
  ],
  "districtCodes": [
    "52222-21"
  ],
  "postalCodes": [
    "48231"
  ],
  "regions": [
    "64"
  ],
  "zones": [
    "6"
  ],
  "coordinates": [
    {
      "latitude": 6.2345,
      "longitude": 50.6789,
      "distance": 5
    }
  ],
  "ags": [
    "18807060"
  ]
```

```
}

HTTP/1.1 200
[
    {
        "id": "64760-11",
        "type": "Feature",
        "properties": {
            "id": "64760-11",
            "centroid": {
                "type": "Point",
                "coordinates": [
                    9.01631706737023,
                    49.536565032187156
                ]
            },
            "hh": 439,
            "zbn": "64760-11"
        },
        "geometry": {
            "type": "MultiPolygon",
            "coordinates": [
                [
                    [
                        [
                            9.049622608052276,
                            49.519275801389085
                        ],
                            .
                            .
                            .multiple coordinates following
                            .
                            .
                        [
                            9.048336576795798,
                            49.51598542686777
                        ]
                    ]
                ]
            ]
        }
```

```
    },
    .
    .
. information on multiple delivery districts following
.
.
]
```

As with the "GET" endpoints, all other data retrieving "POST" endpoints work in the same way as the depicted example. They all return data about the geometric shape and meta information regarding the provided parameters.

Below, you find a list of the available endpoints and information about the data they return.

Endpoint	Data returned
/zone/search	Returns geometric zone information.
/region/search	Returns geometric region information.
/postal-code/search	Returns geometric postal code information.
/neighbourhood/search	Returns geometric neighbourhood information.
/delivery-district/search	Returns geometric delivery-district information.

ID's for partially known data

For the previous described endpoints you need the exact names or ID's to request the geographic information. If you don't know the exact ID's, you can use the following general search endpoint.

/search

With providing the request parameter "query" to this HTTP "GET" method you can search for the needed ID's.

```
GET
https://print-mailing-api.deutschepost.de/targeting/search?query
=kfurt
Accept: application/json
Accept-Encoding: gzip

HTTP/1.1 200
{
  "results": [
    {
      "id": "2888450",
      "areaCode": "Stadt Frankfurt",
      "type": "NEIGHBOURHOOD"
    },
    {
      "id": "Frankfurt am Main",
      "areaCode": "Frankfurt am Main",
      "type": "CITY"
    },
    {
      "id": "14868",
      "areaCode": "Frankfurter Berg",
      "type": "NEIGHBOURHOOD"
    },
    {
      "id": "14866",
      "areaCode": "Frankfurt",
      "type": "NEIGHBOURHOOD"
    },
    {
      "id": "55715",
      "areaCode": "Stadt Frankfurt am Main",
      "type": "NEIGHBOURHOOD"
```

```
    },
    {
      "id": "14867",
      "areaCode": "Frankfurt",
      "type": "NEIGHBOURHOOD"
    },
    {
      "id": "Frankfurt (Oder)",
      "areaCode": "Frankfurt (Oder)",
      "type": "CITY"
    }
  ]
}
```

The parameter has to be URI encoded, so you can search for e.g. "Üb" like "%C3%9Cb". Possible results are postal code, delivery district, city and neighbourhood. The result list is limited to ten items.

Getting shapes with a KML- or Shape-file

For users of the KML- or Shape-file format we offer endpoints for the postal-code and delivery district detail level. Via a multipart file request it os possible to retrieve all intersecting shapes from our service. The intersection percentage can be parameterized via the "threshold".

/files/

The following example uploads a kml file:

```
POST
https://print-mailing-api.deutschepost.de/targeting/postal-code/
files?threshold=50
Content-Type: multipart/form-data; boundary=boundary
Accept: application/json

--boundary
Content-Disposition: form-data; name="file";
filename="Beispiel.kml"
Content-Type: application/xml
```

```
< ../the/file/content...
--boundary

HTTP/1.1 200
[
  {
    "id": "26135",
    "type": "Feature",
    "properties": {
      "id": "26135",
      "centroid": {
        "type": "Point",
        "coordinates": [
          8.262068501492301,
          53.12199657456612
        ]
      },
      "hh": 8763,
      "postal_code": "26135"
    },
    "geometry": {
      "type": "MultiPolygon",
      "coordinates": [
        [
          [
            [
              8.24304468381723,
              53.14197450152309
            ],
            ...
          ]
        ]
      ]
    }
  },
  ...
]
```

Using the playground part of the API to explore the possibility`s with our data

Note that this part of our API is only for exploration and is very different in use from the authenticated part, which is needed for further processing of the selections. There are no cross over points between these API parts at the moment. If you want to store your selection permanently you have to use the authenticated part of the API, which is available for registered and logged in users.

Creating a selection

Now that you have the necessary data for your selection, you need to create a selection ID. For this process it is necessary to preserve the set cookies between requests to ensure the correct routing. This is as easy as calling the HTTP "POST" method:

/selections

In the body of this call you can add several extra parameters to your selection, but for demonstration purposes you are just going to give it a name.

```
POST
https://print-mailing-api.deutschepost.de/targeting/selections
Content-Type: application/json

{
  "name": "Test selection"
}
```

As response you receive a body which informs you about different default values set. The most important out of these values is the "id" value which represents the selection ID that identifies this selection.

```
HTTP/1.1 200
{
    "meta": {
        "id": "afa656db-7c1f-44e7-adb6-74ece5e3627b",
        "name": "Test selection",
        "type": "POSTAL_CODE",
```

```
        "productType": "POSTAKTUELL_ALL",
        "numberOfAreas": 0,
        "netQuantity": 0,
        "quantity": 0,
        "filters": [],
        "filterCosts": 0
    },
    "areaCodes": []
}
```

Right now, your selection only exists in the server cache for a limited time. If you want to store your selection permanently you have to use the authenticated part of the API. Let us move to the next step.

Adding to and removing from a selection

You add data to your selection by calling the HTTP "PATCH" method:

selections/{selection_id}/areacodes

where **{selection_id}** is replaced by your selection ID. In the body, you include the data that you want to be added to your selection. Here, you can use the same kind of information you can use for the search endpoint before.

```
PATCH
https://print-mailing-api.deutschepost.de/targeting/selections/a
fa656db-7c1f-44e7-adb6-74ece5e3627b/areacodes
Content-Type: application/json
Accept: application/json
{
  "add": {
    "cities": [
      {
        "city": "Hamburg",
        "neighbourhood": "Altstadt",
        "latitude": 6.2345,
        "longitude": 50.6789
      }
    ],
    "districtCodes": [
```

```
      "52222-21"
    ],
    "postalCodes": [
      "48231"
    ],
    "regions": [
      "64"
    ]
  }
}

HTTP/1.1 200

{
  "added": [
    {
      "id": "64839",
      "type": "Feature",
      "properties": {
        "id": "64839",
        "centroid": {
          "type": "Point",
          "coordinates": [
            8.873886799926947,
            49.92399923684779
          ]
        },
        "hh": 5504,
        "postal_code": "64839"
      },
      "geometry": {
        "type": "MultiPolygon",
        "coordinates": [
          [
            [
              [
                8.870207983032461,
                49.9058558518063
              ],
                .
                .
```

```
                  .multiple coordinates are following
                  .
                  .
              ]
            ]
          ]
    }, { ... information on multiple delivery districts
following ... }
  ],
  "removed": []
}
```

If you want to remove data from your selection, add the this data to the **"remove"** field instead of the **"add"** field:

```
PATCH
https://print-mailing-api.deutschepost.de/targeting/selections/a
fa656db-7c1f-44e7-adb6-74ece5e3627b/areacodes
Content-Type: application/json
Accept: application/json

{
  "remove": {
    "districtCodes": [
      "52222-21"
    ]
  }
}

HTTP/1.1 200

{
  "added": [],
  "removed": [
    {
      "id": "52222",
      "type": "Feature",
      "properties": {
        "id": "52222",
        "centroid": {
          "type": "Point",
```

```
          "coordinates": [
            6.215589616296115,
            50.78259186178411
          ]
        },
        "hh": 9914,
        "postal_code": "52222"
      },
      "geometry": {
        "type": "MultiPolygon",
        "coordinates": [
          [
            [
              [
                6.196518074681458,
                50.77214738012821
              ]
                .
                .
                .multiple coordinates are following
                .
                .
            ]
          ]
        ]
      }
    }
}
```

Using filters to target specific recipients with POSTWURFSPEZIAL

When using POSTWURFSPEZIAL, you are able to provide filters targetting only specific recipient groups. The quantity of households returned is lowered to the amount available based on your provided filters.

selections/{selection_id}/filters

where **{selection_id}** is replaced by your selection ID. In the body, you include all filters that you want to be used. Filters are changed by sending the full list of all used

filters, therefore using the "PUT" method. This is different from endpoints like **selections/{selection_id}/areacodes**

```
PUT
https://print-mailing-api.deutschepost.de/targeting/selections/4
c436792-5573-47ff-8154-1de446a14319/filters
Content-Type: application/json
Accept: application/json
Accept-Encoding: gzip

[
  {
    "id": "k_kaufkr_kombi",
    "options": ["1"]
  },
  {
    "id": "k_dom_geschlecht_geb",
    "options": ["3", "4"]
  }
]

HTTP/1.1 200

{
  "meta": {
    "id": "c529a31d-41e9-43d6-87f5-fed728608f46",
    "type": "POSTAL_CODE",
    "productType": "POSTWURF_SPEZIAL",
    "numberOfAreas": 3,
    "netQuantity": 17937,
    "quantity": 1233,
    "filters": [
      {
        "id": "k_kaufkr_kombi",
        "options": [
          "1"
        ]
      },
      {
        "id": "k_dom_geschlecht_geb",
        "options": [
```

```
          "3",
          "4"
        ]
      }
    ]
  },
  "areaCodes": [
    "53129",
    "53119",
    "53117"
  ]
}
```

All possible filters with options and description can also be requested via OPTIONS API call below. Available filters are:

ID	name	options
k_kaufkr_kombi	Buying power (weighted)	1. strongly below average 2. below average 3. on bottom of average 4. average 5. on top of average 6. above average 7. strongly above average
k_single_mz	Share of singles	1. extremely low 2. very low 3. low 4. normal 5. high 6. very high 7. extremely high
geb_typ_extern	Building type	1. Detached 1-2 family house 2. Terraced / semi-detached house 3. Small multi-family house, 3-6 HH 4.

		Large multi-family house, larger 6 HH 5. High-rise building
k_garten_reg_geb	Garden size (regionally weighted)	1. no garden 2. small garden 3. normal garden 4. big garden
k_familien_geb	Share of families	1. low 2. normal 3. high 4. very high
m_pers_alter	Age	*18-29 = 18 to 29 years* 30-34 = 30 to 34 years *35-39 = 35 to 39 years* 40-44 = 40 to 44 years *45-49 = 45 to 49 years* 50-54 = 50 to 54 years *55-59 = 55 to 59 years* 60-64 = 60 to 64 years * 65-130 = ab 65 years
k_dom_geschlecht_geb	Share gender	1. Share of men greater than 90% 2. Share of men between 60 % and 90 % 3. Equal distribution between men and women 4. Share of women between 60 % and 90 % 5. Share of women greater than 90 %
k_eigen_geb	Share of ownership	1. low 2. normal 3. high 4. very high

k_bj_v1945_geb	Year of construction before 1945	1. extremely low/not present 2. very low 3. low 4. on bottom of average 5. average 6. on top of average 7. high 8. very high 9. extremely high
k_bj_1945b1974_geb	Year of construction 1945-1974	1. extremely low/not present 2. very low 3. low 4. on bottom of average 5. average 6. on top of average 7. high 8. very high 9. extremely high
k_bj_1975b2000_geb	Year of construction 1975-2000	1. extremely low/not present 2. very low 3. low 4. on bottom of average 5. average 6. on top of average 7. high 8. very high 9. extremely high
k_bj_2001b2005_geb	Year of construction 2001-2005	1. extremely low/not present 2. very low 3. low 4. on bottom of average 5. average 6. on top of average 7. high 8. very high 9. extremely high
k_bj_2006b2010_geb	Year of construction 2006-2010	1. extremely low/not present 2. very low 3. low 4. on bottom of average 5. average 6. on top of average 7. high 8. very high 9. extremely high

k_bj_a2011_geb	Year of construction from 2011	1. extremely low/not present 2. very low 3. low 4. on bottom of average 5. average 6. on top of average 7. high 8. very high 9. extremely high

Show Full Table

Receiving all possible filters with options and description via OPTIONS API call

pws-filter

This API call responds with all available filters, their options and descriptions.

```
OPTIONS
https://print-mailing-api.deutschepost.de/targeting/pws-filter
Accept: application/json
Accept-Encoding: gzip

HTTP/1.1 200

[
  {
    "topic": "k_kaufkr_kombi",
    "name": "Kaufkraft (gewichtet)",
    "description": "Die Kaufkraft-Information liegt auf
Straßenabschnittsebene für durchschnittlich 25 Haushalte vor.
Bei der gewichteten Kaufkraft werden Westdeutschland und
Ostdeutschland getrennt in sieben jeweils gleich große Klassen
unterteilt und dadurch die unterschiedlichen Durchschnittswerte
berücksichtigt.",
    "options": {
      "1": "stark unterdurchschnittlich",
      "2": "unterdurchschnittlich",
      "3": "unterer Durchschnitt",
      "4": "Durchschnitt",
      "5": "oberer Durchschnitt",
      "6": "überdurchschnittlich",
```

```
      "7": "stark überdurchschnittlich"
    }
  },
  {
    "topic": "k_single_mz",
    "name": "Anteil Singles",
    "description": "Auf Basis der Informationen der
Postreferenz-Datei werden Haushalte ermittelt, in denen
wahrscheinlich nur eine Person lebt.",
    "options": {
      "1": "extrem niedrig",
      "2": "sehr niedrig",
      "3": "niedrig",
      "4": "mittel",
      "5": "hoch",
      "6": "sehr hoch",
      "7": "extrem hoch"
    }
  },
  ...
]
```

Reading the selection

Now that you are done selecting, you want to extract the needed information to actually use your selection data. In order to do this, you call the HTTP "GET" method:

selections/{selection_id}/?level={level}

As you can see, the call has a query parameter called "level". Its value can be set to 3 different values:

Level	Data returned
META	Returns meta data of the selection including area codes (default).

DETAIL	Returns detailed data of the selection, including the selection shapes (geodata).

The one value you are using in the following example is the **"META"** parameter. Using this level gives you the following result:

```
GET
https://print-mailing-api.deutschepost.de/targeting/selections/3
cb2c0b7-e96c-4e58-9768-4e6bfcc59285?level=META
Content-Type: application/json
Accept: application/json
Accept-Encoding: gzip

HTTP/1.1 200

{
  "meta": {
    "id": "b725b034-4e38-495e-a861-e0cf463a8119",
    "type": "POSTAL_CODE",
    "productType": "POSTAKTUELL_ALL",
    "numberOfAreas": 64,
    "netQuantity": 303639,
    "quantity": 303639,
    "filters": [],
    "filterCosts": 0
  },
  "areaCodes": [
    "64395",
    "64397",
    "64750",
    "64668",
    "64747",
    .
    .
    multiple postal codes are following
    .
    .
  ]
}
```

With this data you gain an overview of the currently selected postal codes.

Extracting an enumeration from your selection

If you want to evaluate your selection and see how many households will receive a mailing, you can call the HTTP "GET" method:

selections/{selection_id}/dispatch-information

Using this method gives you the following result that can be used in conjunction with the Dispatch-Preparation API:

```
GET
https://print-mailing-api.deutschepost.de/targeting/selections/3
cb2c0b7-e96c-4e58-9768-4e6bfcc59285/dispatch-information
Content-Type: application/json
Accept: application/json
Accept-Encoding: gzip

HTTP/1.1 200

{
  "numberItemsTariffZoneA": 168667,
  "numberItemsTariffZoneB": 134972,
  "itemWeightInGram": 20,
  "productType": "POSTAKTUELL_ALL",
  "deliveryDistrictSelection": false,
  "inductionDate": "2021-06-24T07:43:56.001Z"
}
```

Creating a snapshot of your selection

Your selection is only temporarily saved in the server cache. However, you can get an immutable snapshot of your selection with an extended lifespan by calling:

/selections/{selection_id}/save

where **{selection_id}** is replaced by your selection ID.

```
POST
https://print-mailing-api.deutschepost.de/targeting/selections/afa656db-7c1f-44e7-adb6-74ece5e3627b/save
Content-Type: application/json

HTTP/1.1 200
{
    "id": "afa656db-7c1f-44e7-adb6-74ece5e3627b",
    "expiryDate": [
        2021,
        9,
        7
    ]
}
```

As you can see, you receive a body with a new (semi)permanent ID which represents a read-only copy of your selection, and an expiry date on which your copy will be automatically deleted.

To review this copy of your selection, you can call the HTTP "GET" method:

/permalinks/{permalinkId}

where **{permalinkId}** is replaced by your newly acquired ID.

In case the data basis has changed you get a new "dataVersion" in the response and if your selected areas changed you find this information in the "areaCodeHouseholdDifferences" list.

```
GET
https://print-mailing-api.deutschepost.de/targeting/permalinks/afa656db-7c1f-44e7-adb6-74ece5e3627b

HTTP/1.1 200
{
    "meta": {
        "id": "afa656db-7c1f-44e7-adb6-74ece5e3627b",
        "type": "POSTAL_CODE",
        "productType": "POSTAKTUELL_ALL",
        "numberOfAreas": 0,
        "netQuantity": 0,
```

```
        "quantity": 0,
        "dataVersion": "2011-09-01T23:56:99",
        "filters": [],
        "filterCosts": 0
    },
    "areaCodes": [
        "64395",
        "64397",
        "64750",
        .
        .
        . multiple area codes following
        .
        .
    ],
    "areaCodeHouseholdDifferences": [
      {
        "areaCode": "64395",
        "households": -1
      }
    ]
}
```

If your original selectionId is no longer in the application's cache, you can reload it for further editing. To do this, you can call the HTTP "POST" method:

/selections/load?permalinkId={permalinkId}

where **{permalinkId}** is replaced by the ID you received when saving your selection.

In case the data basis has changed you get a new "dataVersion" in the response and if your selected areas changed you find this information in the "areaCodeHouseholdDifferences" list. Please note only the new "dataVersion" is available for further editing of your selection.

```
POST
https://print-mailing-api.deutschepost.de/targeting/selections/l
oad?permalinkid=afa656db-7c1f-44e7-adb6-74ece5e3627b
Content-Type: application/json
Accept: application/json
Accept-Encoding: gzip
```

```
HTTP/1.1 200
{
    "meta": {
        "id": "7475f415-6806-45d1-81d8-84c1145dcd3d",
        "type": "POSTAL_CODE",
        "productType": "POSTAKTUELL_ALL",
        "numberOfAreas": 0,
        "netQuantity": 0,
        "quantity": 0,
        "dataVersion": "2011-09-01T23:56:99",
        "filters": [],
        "filterCosts": 0
    },
    "areaCodes": [
        "64395",
        "64397",
        "64750",
        .
        .
        . multiple area codes following
        .
        .
    ],
    "areaCodeHouseholdDifferences": [
      {
        "areaCode": "64395",
        "households": -1
      }
    ]
}
```

With the new ID you can now use all **/selection/{selectionId}** requests again. The save and load requests for one id are limited to three requests per hour to prevent abuse.

And that is it. These are the core concepts of this API. For more specific information on the requests, please refer to the OpenAPI definitions.

Using the authenticated part of the API (this is required to proceed to booking or planning)

Everything in the following endpoints is tied to a customer id, therefore the endpoints are starting with: **/targeting/customers/{customer_id}/** To use these resources you need to provide a JWT with the adequate access rights and customer rights.

Listing all selections of a customer

With the following request you can list all selections of a customer:

/targeting/customers/{customer_id}/selections

```
GET
https://print-mailing-api.deutschepost.de/targeting/customers/1/
selections
Content-Type: application/json
Accept: application/json

HTTP/1.1 200
[
  {
    "id": "1ab4c485-535f-474f-8dd3-4bf005bd3db2",
    "name": "foo"
  },
  {
    "id": "51be899e-ca6f-43a4-b8f1-a659a773fdea",
    "name": "bar"
  }
]
```

Get a specific selection

With the following request you can get a specific selection of a customer:

/targeting/customers/{customer_id}/selections/{selection_id}

The normal behavior of this request is to simply return the saved values from the DB, but this request has an optional url parameter "recalculate" which can be set to true, to trigger a recalculation of all values based on the selected areas, occurring changes show in the filed "areaCodeHouseholdDifferences". We recommend the use of the "recalculation" flag to assure the data response contains the newest data.

```
GET
https://print-mailing-api.deutschepost.de/targeting/customers/1/
selections/51be899e-ca6f-43a4-b8f1-a659a773fdea
Content-Type: application/json
Accept: application/json

HTTP/1.1 200
{
  "meta": {
    "id": "51be899e-ca6f-43a4-b8f1-a659a773fdea",
    "customerId": "11",
    "name": "bar",
    "type": "POSTAL_CODE",
    "productType": "POSTAKTUELL_DAILY_MAIL",
    "numberOfAreas": 1,
    "netQuantity": 18542,
    "quantity": 12926,
    "itemWeightInGram": 20,
    "dataVersion": "2011-09-01T23:56:99",
    "filters": [],
    "filterCosts": 0
  },
  "areaCodes": [
    "31535"
  ],
  "areaCodeHouseholdDifferences": []
}
```

Save a new selection for a customer

To save a new selection you can use the following url as a POST:

/targeting/customers/{customer_id}/selections

```
POST
https://print-mailing-api.deutschepost.de/targeting/customers/1/
selections
Content-Type: application/json
Accept: application/json

{
  "name": "Fooo",
  "areaCodes": [
    "31535"
  ],
  "detailLevel": "POSTAL_CODE",
  "product": "POSTAKTUELL_DAILY_MAIL",
  "filters": [],
  "weight": "20"
}

HTTP/1.1 200
{
  "id": "41d709d6-3a96-41e2-89d6-da51a4a07f96",
  "name": "Fooo"
}
```

The return value contains the {selection_id} which is the main identifier for other processes with this selection.

Change a selection

A selection can be changed by its id, but note that the customer_id can not be changed.

/targeting/customers/{customer_id}/selections/{selection_id}

```
PATCH
https://print-mailing-api.deutschepost.de/targeting/customers/1/
selections/41d709d6-3a96-41e2-89d6-da51a4a07f96
Content-Type: application/json
Accept: application/json

{
```

```
  "name": "FooBar",
  "areaCodes": [
    "20257"
  ],
  "detailLevel": "POSTAL_CODE",
  "product": "POSTAKTUELL_DAILY_MAIL",
  "filters": [],
  "weight": "20"
}

HTTP/1.1 200
{
  "id": "41d709d6-3a96-41e2-89d6-da51a4a07f96",
  "name": "FooBar"
}
```

If you change "detailLevel" of the selection you have to adjust the areaCodes to the new value, areaCodes which are not found within the new "detailLevel" will be removed. If you change away from the product "POSTWURF_SPEZIAL" and you have filters defined you'll get an error.

Rename a selection

A selection can be renamed by its id.

/targeting/customers/{customer_id}/selections/{selection_id}/name

```
PATCH
https://print-mailing-api.deutschepost.de/targeting/customers/1/
selections/41d709d6-3a96-41e2-89d6-da51a4a07f96/name
Accept: application/json

Subar

HTTP/1.1 200
{
  "id": "41d709d6-3a96-41e2-89d6-da51a4a07f96",
  "name": "Subar"
}
```

Get a dispatch information preview of the data from a selection

A selection can be deleted by its id.

/targeting/customers/{customer_id}/selections/{selection_id}/dispatch-information

```
GET
https://print-mailing-api.deutschepost.de/targeting/customers/1/selections/41d709d6-3a96-41e2-89d6-da51a4a07f96/dispatch-information
Accept: application/json

HTTP/1.1 200
{
  "numberItemsTariffZoneA": 0,
  "numberItemsTariffZoneB": 1695,
  "itemWeightInGram": 20,
  "productType": "POSTAKTUELL_DAILY_MAIL",
  "deliveryDistrictSelection": false,
  "inductionDate": "2021-12-21T12:57:42.001Z"
}
```

Delete a selection

A selection can be deleted by its id.

/targeting/customers/{customer_id}/selections/{selection_id}

```
DELETE
https://print-mailing-api.deutschepost.de/targeting/customers/1/selections/41d709d6-3a96-41e2-89d6-da51a4a07f96/name
Accept: application/json

HTTP/1.1 200
```

Legal Terms

Legal Terms for the use of and/or access to the Print Mailing Targeting API

These Legal Terms do not replace and/or modify the dispatch conditions for the products Dialogpost, Dialogpost schwer, Postwurfspezial and Postaktuell pursuant to the respectively valid service brochures and associated general terms and conditions and the Terms of Use for the Order Management System of Deutsche Post AG, as well as of Deutsche Post Direkt or any other mail and/or shipment services agreement, which govern Your Deutsche Post or Deutsche Post Direkt shipments and mailings.

Only PDF documents created directly from the Print Mailing Targeting API may be used for posting. You are not permitted to program your own posting lists.

We offers no warranty with respect to the contents and results that can be obtained from the use of the API (hereinafter also referred to as “Service”), or the accuracy and reliability of the information received within the Service, and we do not guarantee that the quality of the information received in connection with the Service will match your or your customers’ expectations. In particular, the information in the data may differ from the shipment quantities, weights and formats actually posted and therefore simply reflects a status before the shipment was posted.

As a result of regular database updates, there may be differences between the number of addresses stated here and the number of addresses upon delivery.

You may use the data collected via the Print Mailing Targeting API for the purposes specified in these Legal Terms, the General Developer Portal Terms of Use, the API descriptions and only as permissible according to the relevant laws.

Support

Should you require assistance in connecting the Print-Mailing Targeting API, please contact our Support Team directly at the e-mail address it-csp@deutschepost.de with mail subject "[#tar] ".

NOTE: Please contact us if you recognize errors, want to improve the documentation or have questions!

You will promptly receive a confirmation with a ticket number. Reply directly to that e-mail should you have questions or additional information. Do not change the subject header containing the ticket number so that we can correctly match further messages with your initial query.

Please use this e-mail address exclusively so that we can process your query as quickly as possible. Our Support Team will assist you with your questions and refer you to additional experts if needed. Please understand that we cannot answer any development-related questions (e.g. code generation in dedicated programming languages).

Reference Docs

Download API Spec

The Target Group Selection API enables the user to make a selection of delivery districts based on addresses and/or coordinates, and thus configure the best suited delivery area for his advertising materials.

Contact Technical Support

Apache 2.0

Servers

https://uat.print-mailing-api-test.deutschepost.de/targeting - UAThttps://print-mailing-api.deutschepost.de/targeting - PROD

Selection

This Authenticated And Customer Specific Selection Endpoints Can Be Used To Prepare A Selection For Further Use (Store, Booking ...). All Requests Need Authentication Via JWT.

Endpoints

`GET`

`/customers/{customerId}/selections`

List selection of a specific customer

`POST`

`/customers/{customerId}/selections`

Create a new selection for a customer

GET

/customers/{customerId}/selections/{selectionId}

Get a stored selection by the id for a specific customer

DELETE

/customers/{customerId}/selections/{selectionId}

Deletes a selection

PATCH

/customers/{customerId}/selections/{selectionId}

Update a selection

PATCH

/customers/{customerId}/selections/{selectionId}/name

Updates the name of a selection

GET

/customers/{customerId}/selections/{selectionId}/packaging

Get information's for packaging of the selection

GET

/customers/{customerId}/selections/{selectionId}/dispatch-information

Request a summary about the information to dispatch the selection

Lookups

Lists Static Request Oder Response Values Of Other API Resources

Endpoints

OPTIONS

/pws-filter

GET

/errorcode-lookups

Temporary Selection

This Temporary Selection Can Be Used To Explore The Data Provided By This API, But Not To Directly Proceed To Order A Print Mailing. For These Requests, It Is Necessary To Preserve The Set Cookies Between Requests To Ensure The Correct Routing.

Endpoints

PUT

/selections/{selectionId}/filters

Update a selections filters.

POST

/selections

Creates a new selection inside the cache

POST

/selections/{selectionId}/save

POST

/selections/{selectionId}/areacodes/files

Update a selection with a file upload.

POST

/selections/load

PATCH

/selections/{selectionId}/product

Update a selection detail level.

PATCH

/selections/{selectionId}/detaillevel

Update a selection detail level.

PATCH

/selections/{selectionId}/areacodes

Update a selection.

GET

/selections/{selectionId}

Returns a selection by ID from cache

DELETE

/selections/{selectionId}

Removes a selection from cache

GET

/selections/{selectionId}/pam-export

GET

/selections/{selectionId}/dispatch-information

Data

Endpoints

POST

/zone/search

Get a list of zone information by a zone search.

POST

/region/search

POST

/postal-code/search

POST

/postal-code/files

Get a list of shapes with a shape file in zip format or a kml file upload.

POST

/postal-code/costs

POST

/neighbourhood/search

POST

/neighbourhood/search/{neighbourhood}

POST

/neighbourhood/costs

POST

/delivery-district/search

POST

/delivery-district/files

Get a list of shapes with a shape file in zip format or a kml file upload.

POST

/delivery-district/costs

GET

/zone

Get geometric information for all zones.

GET

/zone/{zone}

GET

/search

GET

/region/{region}

GET

/postal-code/{postal-code}

GET

/neighbourhood/{neighbourhood}

GET

/delivery-district/{delivery-district}

GET

/delivery-district/search/{query}

Permalink

Endpoints

POST

/permalinks

Saves a new selection as permalink in the database

GET

/permalinks/{permalinkId}

Get the persistent selection of the permalink id

Downloads

By downloading any file, you accept our **Terms of Use**

Open API Specification File
YAML