Rundfunk & Telekom
Regulierungs-GmbH

RTR

RTR Opendata REST documentation

RTR-GmbH

Version 1.0.0, 2016-11-03

Table of Contents

1. Common

This document outlines the usage of RTR Opendata information and the corresponding API.
A description of properties common to all endpoints is followed by individual, more in depth explanations.
Occurrences of `<>` need to be replaced by their actual values.

1.1. Base url

The RTR Opendata API is available at `https://data.rtr.at/api/v1/tables`
(from now on referred to as `<base_url>`).

1.2. Response formats

All endpoints support returning data in JSON (default if nothing specified) or in XML format. The single table data endpoint additionally allows for CSV formatting.
The response's format can be controlled by appending the request url with `.json`, `.xml` or `.csv` for JSON, XML or CSV respectively.
However, in general it will have `status`, `message`, `timestamp` and `data` fields, where
`status` is set to the http status code
`message` is a human readable string describing the status
`timestamp` is the date/time the server processed the request (formatted as ISO 8601)
`data` additional response data
`version` some endpoints add the optional version-field, specifying which version of the data set was (automatically) selected. If no version for this table is available this will be null, otherwise contains an object with `id` (version number - an integer) and `published` fields (datetime the data was released - formatted as ISO 8601)

1.2.1. JSON

```
{
  "message" : "OK",
  "status" : 200,
  "timestamp" : "2016-07-01T10:00:00+02:00",
  "data" : [..]
}
```

1.2.2. XML

Will always return a root object `ResponseWrapper` with the fields as child nodes.
If the data-field contains an array value, each entry will be encapsulated by a separate `data`-node:

```
<ResponseWrapper>
  <message>OK</message>
  <status>200</status>
  <timestamp>2016-07-01T10:00:00+02:00/timestamp>
  <data>
    ..
  </data>
  <data>
    ..
  </data>
  ..
</ResponseWrapper>
```

1.3. Response status codes

Successful requests return a **200** http status code and the requested data.

Should a problem occur whilst processing the request, a non-200 status code is returned together with a JSON object containing the status code, a string description and the timestamp (see above).

Specifically:
400 status code: The user made a bad request to the server (e.g. invalid parameter)
404 status code: A resource (e.g. table) or path could not be found
503 status code: The client has made too many requests to the REST-interface and has exceeded the rate limit

1.4. Request parameters

Should an endpoint accept parameters further specifying the desired data, shall they be provided by optionally adding a query string (with name/value pairs) to the `GET` request.
Such a string has the form: `?name_1=value_1(&<name_x>=<value_x>)*`
where <name_x> specifies the parameter's name and <value_x> its content. The syntax `(..)*` means that everything encompassed by the parenthesis may be repeated 0 or more times.
Further, should the url and its values be url-encoded to escape special characters.

1.4.1. Pagination

Pagination of result entries can be achieved by providing the `page` and `size` parameters.

Page

The page filter selects which page should be shown.
Valid values are 0 or greater.
Default (if not present): 0
E.g. to select the first page set this filter to 0.

NOTE Requires the size filter to be present and greater than 0

Size

The size filter defines the maximum number of objects that should be returned by a single request (on a single page).
Valid values are 0 or greater.
Default (if not present): 0
A value of 0 means that no size limit will be applied and all available entries will be returned (e.g. for download). The page filter will be ignored in this case.

Example

Default request (page 0, size 0) - returns all available objects from table `plz`

```
$wget https://data.rtr.at/api/v1/tables/plz
```

equal to

```
$wget https://data.rtr.at/api/v1/tables/plz?size=0&page=0
```

Page/Size request (page 1, size 25) - returns up to 25 objects on the second page
(Objects 26-50)

```
$wget https://data.rtr.at/api/v1/tables/plz?page=1&size=25
```

Custom parameters

Any other name (except for `page` and `size`) may be used as a parameter as long as they are listed under `columns` in the description endpoint.

The default operation to compare values is the exact match.
E.g. if one were to filter for `<field>` = `<value>`, she would receive only elements where `<field>` is equal to `<value>` (alias for **eq_** - see below)
However, it is possible to use other methods of comparison by applying a prefix:

eq_ Filter all values that are precise matches to `<value>`
(Default - `eq_<parameter>` is identically to the no prefix `<parameter>`)
neq_ Filter all values that are not precise matches to `<value>`
null_ Filter all values that are `null`.
`value` should not be present. (e.g. only `null_<field>` or `null_<field>=`)
nnull_ Filter all values that are not `null`.
`value` should not be present. (e.g. only `nnull_<field>` or `nnull_<field>=`
lt_ Filter all values lower than `<value>`
lte_ Filter all values lower than or equal to `<value>`
gt_ Filter all values greater than `<value>`
gte_ Filter all values greater than or equal to `<value>`
lk_ Filter all values like `<value>` (Wildcards may be set by including `*`)

Filtering is case-insensitive.
E.g. A value of `B*e` would match `Be`, `Bee`, `Binge` or `bee`, but not `Bees`

If multiple filters with different prefixes are present, they are logically interconnected by ANDs.
E.g. By supplying `gt_count=5` and `lt_count=8`, one would receive all entries where `count` ranges between 5 and 8 (either 6 or 7).

Example

Filter table `plz` on columns `plz` greater than 2000 and less than 3000

```
$wget https://data.rtr.at/api/v1/tables/plz?gte_plz=2000&lt_plz=3000
```

2. Endpoints

Description of available endpoints for RTR Opendata REST API.

2.1. Available tables

Url: `<base_url>`
Format: `<base_url>.json` , `<base_url>.xml`
Parameter: None
Description: Shows a list of all available tables the REST interface is aware of.

2.1.1. Example

Request

```
$wget https://data.rtr.at/api/v1/tables/
```

Response

```
{
  "message" : "OK",
  "status" : 200,
  "timestamp" : "2016-11-03T14:08:35+01:00",
  "data" : ["MedKFTGAmpelliste","MedKFTGBekanntgabe","MedienAGG",...,"skp","tkSTS",
"tk_frequenzen"]
}
```

2.2. Single table

Url: `<base_url>/<name>` (`<table_url>`)
Where `<name>` is the name of a table returned by the available tables endpoint.

2.2.1. Description

Url: `<table_url>/description`
Format: `<base_url>/<name>/description.json` , `<base_url>/<name>/description.xml`
Parameter: Language selection
The language of the response data may be requested by providing a `Accept-Language` header or by supplying a `lang` url parameter.

NOTE A url parameter, if present, will supersede the `Accept-Language` header

Description: Retrieve general and translated information for this data set (Meta information, available columns, ..)

Response fields

metadata: List of meta informations for this table
Specific values depend on the data set, but are in general objects with two fields:

- "name": Designation of this meta information
- "content": Content/Value of this meta information

columns: List of available data columns:
Specific values depend on the data set, but are of the general shape `"key": column_record`.
These `<key>` 's may be used for filtering where applicable as specified in custom parameters.
A single column record contains two fields:

- "name": Translated column name
- "description": Translated column description

table: Translated table name
description: Translated table description

Example

Request

```
$wget https://data.rtr.at/api/v1/tables/plz/description
```

Response

```
{
  "message": "OK",
  "status": 200,
  "timestamp": "2016-10-19T16:14:08+02:00",
  "data": {
    "metadata": [
      {
        "name": "Kategorie",
        "content": "Verkehr und Technik, Verwaltung und Politik"
      },
...
      {
        "name": "Datenverantwortliche Stelle - E-Mailkontakt",
        "content": "rtr@rtr.at"
      }
    ],
    "columns": {
      "plz": {
        "name": "plz",
        "description": "Numerische Bezeichnung der Postleitzahl"
      },
...
      "postfach": {
        "name": "postfach",
        "description": "mögliche Adressierung von Postfächern der PLZ"
      }
    },
    "table": "Postleitzahlen"
  }
}
```

2.2.2. Data

Url: `<table_url>`
Format: `<base_url>/<name>.json` , `<table_url>/<name>.xml` , `<base_url>/<name>.csv`
Parameter: All parameters as described in the requests parameters section
Description: Show data for this table

Example

Get all information

Request

```
$wget https://data.rtr.at/api/v1/tables/plz/
```

Response

```
{
  "message": "OK",
  "status": 200,
  "timestamp": "2016-11-03T14:14:30+01:00",
  "data": [
    {
      "plz": 1000,
      "ort": "Wien",
      "bundesland": "W",
      "gueltigab": "2009-08-01",
      "gueltigbis": null,
      "plztyp": "PLZ-Postfach",
      "internextern": "extern",
      "adressierbar": "Nein",
      "postfach": "Ja"
    },
    {
      "plz": 1004,
      "ort": "Wien",
      "bundesland": "W",
      "gueltigab": "1966-01-01",
      "gueltigbis": null,
      "plztyp": "InteressentenPLZ",
      "internextern": "extern",
      "adressierbar": "Nein",
      "postfach": "Ja"
    },
...
    {
      "plz": 9992,
      "ort": "Iselsberg-Stronach",
      "bundesland": "T",
      "gueltigab": "2009-07-01",
      "gueltigbis": null,
      "plztyp": "PLZ-Adressierung",
      "internextern": "extern",
      "adressierbar": "Ja",
      "postfach": "Nein"
    }
  ],
  "version": {
    "id": 132,
    "published": "2016-08-23T17:07:00+02:00"
  }
}
```

Example

Get all data where `plz=2424`

Request

```
$wget https://data.rtr.at/api/v1/tables/plz?plz=2424
```

Response

```
{
  "message": "OK",
  "status": 200,
  "timestamp": "2016-11-03T14:18:34+01:00",
  "data": [
    {
      "plz": 2424,
      "ort": "Zurndorf",
      "bundesland": "B",
      "gueltigab": "1966-01-01",
      "gueltigbis": null,
      "plztyp": "PLZ-Adressierung",
      "internextern": "extern",
      "adressierbar": "Ja",
      "postfach": "Ja"
    }
  ],
  "version": {
    "id": 132,
    "published": "2016-08-23T17:07:00+02:00"
  }
}
```

2.2.3. Versions

Url: `<table_url>/versions`
Format: `<base_url>/<name>/versions.json` , `<base_url>/<name>/versions.xml`
Parameter: None
Description: List available version information

Response fields

List of objects with values:
idVersion: ID of version
published: Publish date

Example

Request

```
$wget https://data.rtr.at/api/v1/tables/plz/versions
```

Response

```
{
  "message": "OK",
  "status": 200,
  "timestamp": "2016-11-03T14:19:38+01:00",
  "data": [
    {
      "idVersion": "132",
      "published": "2016-08-23T17:07:00+02:00"
    }
  ],
  "version": {
    "id": 132,
    "published": "2016-08-23T17:07:00+02:00"
  }
}
```

2.2.4. Distinct

Url: `<table_url>/distinct`
Format: `<base_url>/<name>/distinct.json` , `<table_url>/<name>/distinct.xml`
Parameter: By providing one or more parameter without a value queries the interface for all distinct values for this column (`?name_1(&name_x)`).
All parameters as described in the custom parameters section may be used to limit and filter the results.
Description: Show unique values for this table and filter settings

NOTE By requesting `page` with a `size` greater than 0, the endpoint returns the number of pages this data set would fill in regards to pagination.

Example

Get all distinct values for `bundesland` and `plztyp`

Request

```
$wget https://data.rtr.at/api/v1/tables/plz/distinct?bundesland&plztyp
```

Response

```
{
  "message": "OK",
  "status": 200,
  "timestamp": "2016-11-03T14:20:50+01:00",
  "data": {
    "plztyp": [
      "FeldPLZ",
      "InteressentenPLZ",
      "PLZ-Adressierung",
      "PLZ-Historisch",
      "PLZ-Postfach"
    ],
    "bundesland": [ "B", "K", "N","O", "Sa", "St", "T", "V", "W" ]
  },
  "version": {
    "id": 132,
    "published": "2016-08-23T17:07:00+02:00"
  }
}
```

Example

Get all distinct values for `plz` and `ort` where `plz>9990`

Request

```
  $wget https://data.rtr.at/api/v1/tables/plz/distinct?ort&gt_plz=9990
```

Response

```
{
  "message": "OK",
  "status": 200,
  "timestamp": "2016-11-03T14:29:01+01:00",
  "data": {
    "ort": [
      "Dölsach",
      "Iselsberg-Stronach"
    ],
    "plz": [
      9991,
      9992
    ]
  },
  "version": {
    "id": 132,
    "published": "2016-08-23T17:07:00+02:00"
  }
}
```

Example

By requesting the distinct value for `page` whilst setting `size=1` one can find out how many data sets are available for this table.
To get the total count of records of table `plz` check the `page` value of the result.

Request

```
  $wget https://data.rtr.at/api/v1/tables/plz/distinct?size=1&page
```

Response

```
{
  "message": "OK",
  "status": 200,
  "timestamp": "2016-11-03T14:31:50+01:00",
  "data": {
    "page": [
      2652
    ]
  },
  "version": {
    "id": 132,
    "published": "2016-08-23T17:07:00+02:00"
  }
}
```

Example

By applying one or more filter expressions to the previous example one can find out how many data sets are available for a filtered table.
Get the count of records of table `plz` where `adressierbar=Ja`

Request

```
  $wget https://data.rtr.at/api/v1/tables/plz/distinct?size=1&page&adressierbar=Ja
```

Response

```
{
  "message": "OK",
  "status": 200,
  "timestamp": "2016-11-03T14:33:55+01:00",
  "data": {
    "page": [
      2218
    ],
    "adressierbar": [
      "Ja"
    ]
  },
  "version": {
    "id": 132,
    "published": "2016-08-23T17:07:00+02:00"
  }
}
```