

# Package ‘ODataQuery’

May 7, 2026

**Type** Package

**Title** Querying on 'OData'

**Version** 0.5.3

**Description** Make querying on 'OData' easier. It exposes an 'ODataQuery' object that can be manipulated and provides features such as selection, filtering and ordering.

**License** GPL-3

**Encoding** UTF-8

**LazyData** false

**Imports** R6, httr, jsonlite, rlang

**RoxygenNote** 7.1.1

**Suggests** knitr, rmarkdown, tinytest, utils

**VignetteBuilder** knitr

**NeedsCompilation** no

**Author** Laurent Verweijen [aut, cre]

**Maintainer** Laurent Verweijen <lauerund+github@gmail.com>

**Repository** CRAN

**Date/Publication** 2021-07-05 09:30:02 UTC

## Contents

and_query . . . . .	2
ODataQuery . . . . .	3
odata_function . . . . .	11
retrieve_all . . . . .	12
retrieve_data . . . . .	13
retrieve_one . . . . .	14
to_odata . . . . .	15

<b>Index</b>	<b>16</b>
--------------	-----------

---

and_query	<i>Create a combined filter</i>
-----------	---------------------------------

---

**Description**

Create a combined filter

**Usage**

```
and_query(...)
```

```
or_query(...)
```

```
not_query(...)
```

**Arguments**

...                      Raw odata queries or query options.

**Details**

This function can be used with raw values or query options

1. Raw odata queries Raw OData can be passed as string. It's the responsibility of the caller that the argument is valid syntax and values are escaped.
2. Query options Query options can be passed as named parameters.

Query options should be of the following form: `property.operator = value`

- Property should be a property of the entity or individual.
- Operation can have any of the following values:
  - eq Whether property is equal to value.
  - ne Whether property is not equal to value.
  - gt Whether property is greater than value.
  - ge Whether property is greater than or equal to value.
  - lt Whether property is lower than value.
  - le Whether property is lower than or equal to value.
  - has Whether property has value as enumeration property.
  - startswith Whether property starts with value.
  - endswith Whether property ends with value.
  - contains Whether property contains value.
- Value should be a string, double or boolean and will be escaped automatically.

**See Also**

<https://docs.oasis-open.org/odata/odata/v4.0/errata03/os/complete/part2-url-conventions/>

## Examples

```
and_query("Column eq OtherColumn",
    FirstName.startswith = 'A',
    LastName.eq = 'Scott')

or_query("ExpireDate eq null",
    ExpireDate.lt = "2020-07-07")

not_query(or_query(Age.lt = 21, Age.gt = 65))
```

---

ODataQuery

*ODataQuery*

---

## Description

R6 class that represents an OData query

## Details

This class has methods to build and navigate OData services:

- Use methods such as `$path()` and `$get()` to find a path.
- Use methods such as `$select()` and `$filter()` to make your query.
- Use methods such as `$retrieve()`, `$all()` and `$one()` to obtain the results.

## Active bindings

`url` Generate (encoded) url

## Methods

### Public methods:

- `ODataQuery$new()`
- `ODataQuery$print()`
- `ODataQuery$path()`
- `ODataQuery$get()`
- `ODataQuery$func()`
- `ODataQuery$query()`
- `ODataQuery$top()`
- `ODataQuery$skip()`
- `ODataQuery$select()`
- `ODataQuery$filter()`
- `ODataQuery$expand()`
- `ODataQuery$orderby()`

- `ODataQuery$search()`
- `ODataQuery$compute()`
- `ODataQuery$retrieve()`
- `ODataQuery$all()`
- `ODataQuery$one()`

**Method** `new()`: Create a class representing a query.

*Usage:*

```
ODataQuery$new(
  service,
  .resource = "",
  .query_options = list(),
  htr_args = list()
)
```

*Arguments:*

`service` The url of the endpoint to connect to. This url should not end with backslash.

`.resource` Should not be used directly. Use `$path()` instead.

`.query_options` Should not be used directly. Use methods such as `$select()`, `$filter()` and `$query()` instead.

`htr_args` Additional parameters to pass to `httr::GET`

value Read-only

*Examples:*

```
service <- ODataQuery$new("https://services.odata.org/V4/TripPinServiceRW")
```

**Method** `print()`: Print query, useful when debugging.

*Usage:*

```
ODataQuery$print(top = 0, ...)
```

*Arguments:*

`top` Number of records to print.

`...` Additional parameters are passed to print

*Examples:*

```
\dontrun{
service <- ODataQuery$new("https://services.odata.org/V4/TripPinServiceRW")
service$print(10)$path("People")$print()
}
```

**Method** `path()`: Supply path to the resource

*Usage:*

```
ODataQuery$path(...)
```

*Arguments:*

`...` Components that lead to resource path

*Examples:*

```
service <- ODataQuery$new("https://services.odata.org/V4/TripPinServiceRW")
people_entity <- service$path("People")
```

**Method** `get()`: Query an individual record by ID parameters

*Usage:*

```
ODataQuery$get(...)
```

*Arguments:*

... ID-parameters (named or unnamed)

*Examples:*

```
service <- ODataQuery$new("https://services.odata.org/V4/TripPinServiceRW")
people_entity <- service$path("People")
russellwhyte <- people_entity$get("russellwhyte")
```

**Method** `func()`: Path to an OData function

*Usage:*

```
ODataQuery$func(fname, ...)
```

*Arguments:*

fname Name of the function

... Options passed to `retrieve_data`

*Returns:* closure

*Examples:*

```
service <- ODataQuery$new("https://services.odata.org/V4/TripPinServiceRW")
get_nearest_airport <- service$func('GetNearestAirport')
\dontrun{
get_nearest_airport(lat = 33, lon = -118)
}
```

**Method** `query()`: Supply custom query options that do not start with \$

*Usage:*

```
ODataQuery$query(...)
```

*Arguments:*

... Named lists where the names are custom query options

*Examples:*

```
service <- ODataQuery$new("https://services.odata.org/V4/TripPinServiceRW")
people_entity <- service$path("People")
people_entity$query(filter = "FirstName eq 'scott')$url
```

**Method** `top()`: Limit the number of results to n

*Usage:*

```
ODataQuery$top(n = 10)
```

*Arguments:*

n Number of records to return at most

*Examples:*

```
service <- ODataQuery$new("https://services.odata.org/V4/TripPinServiceRW")
people_entity <- service$path("People")
people_entity$top(10)
```

**Method skip():** Skip first few items*Usage:*

```
ODataQuery$skip(n = 10)
```

*Arguments:*

n Number of items to skip

*Examples:*

```
service <- ODataQuery$new("https://services.odata.org/V4/TripPinServiceRW")
people_entity <- service$path("People")
people_entity$skip(10)
```

**Method select():** Select fields. If not present, all fields are returned.*Usage:*

```
ODataQuery$select(...)
```

*Arguments:*

... Fields to select

*Examples:*

```
service <- ODataQuery$new("https://services.odata.org/V4/TripPinServiceRW")
people_entity <- service$path("People")
people_entity$select("FirstName", "LastName")
```

**Method filter():** Apply filter to result*Usage:*

```
ODataQuery$filter(...)
```

*Arguments:*

... Can be a raw odata query or query options. It's recommended to use query options because these will automatically escape parameters. The parameters are passed on to and\_query.

*Examples:*

```
service <- ODataQuery$new("https://services.odata.org/V4/TripPinServiceRW")
people_entity <- service$path("People")
people_entity$filter(FirstName.eq = 'Scott')
```

**Method expand():** Expand on expansion properties*Usage:*

```
ODataQuery$expand(...)
```

*Arguments:*

... Properties to extend on

*Examples:*

```
service <- ODataQuery$new("https://services.odata.org/V4/TripPinServiceRW")
people_entity <- service$path("People")
people_entity$expand("Friends")
```

**Method** `orderBy()`: Order results by one or more keys

*Usage:*

```
ODataQuery$orderBy(...)
```

*Arguments:*

... Keys to order by. To order in descending order, the key can be prefixed by a negative sign.

*Examples:*

```
service <- ODataQuery$new("https://services.odata.org/V4/TripPinServiceRW")
people_entity <- service$path("People")
people_entity$orderBy('Concurrency')
people_entity$orderBy('-Concurrency')
```

**Method** `search()`: Search the entity

*Usage:*

```
ODataQuery$search(s)
```

*Arguments:*

s Search string as defined by the endpoint.

*Examples:*

```
service <- ODataQuery$new("https://services.odata.org/V4/TripPinServiceRW")
people_entity <- service$path("People")
people_entity$search('Boise')
```

**Method** `compute()`: Compute properties

Add additional properties to query computed from other attributes.

*Usage:*

```
ODataQuery$compute(...)
```

*Arguments:*

... Named list of properties to compute

*Examples:*

```
# Not really supported by this particular service.
service <- ODataQuery$new("https://services.odata.org/V4/TripPinServiceRW")
people_entity <- service$path("People")
people_entity$compute(a = "5 MUL Concurrency")
```

**Method** `retrieve()`: Retrieve data

*Usage:*

```
ODataQuery$retrieve(count = FALSE, ...)
```

*Arguments:*

count Whether to include a count of the total number of records

... Passed to `retrieve_data`

*Examples:*

```
\dontrun{
service <- ODataQuery$new("https://services.odata.org/V4/TripPinServiceRW")
people_entity$retrieve()
}
```

**Method all():** Retrieve all data pages  
Return concatenation of value of all pages

*Usage:*

```
ODataQuery$new(...)
```

*Arguments:*

```
... Passed to retrieve_all
```

*Examples:*

```
\dontrun{
service <- ODataQuery$new("https://services.odata.org/V4/TripPinServiceRW")
people_entity$all()
people_entity$all(jsonlite_args = list(simplifyVector = False))
}
```

**Method one():** Retrieve individual

*Usage:*

```
ODataQuery$new(...)
```

*Arguments:*

```
... Passed to retrieve_one
```

*Examples:*

```
\dontrun{
service <- ODataQuery$new("https://services.odata.org/V4/TripPinServiceRW")
people_entity$top(1)$one(default = NA)
}
```

**See Also**

[and\\_query\(\)](#) for details.

**Examples**

```
## -----
## Method `ODataQuery$new`
## -----

service <- ODataQuery$new("https://services.odata.org/V4/TripPinServiceRW")

## -----
## Method `ODataQuery$print`
## -----
```

```

## Not run:
service <- ODataQuery$new("https://services.odata.org/V4/TripPinServiceRW")
service$print(10)$path("People")$print()

## End(Not run)

## -----
## Method `ODataQuery$path`
## -----

service <- ODataQuery$new("https://services.odata.org/V4/TripPinServiceRW")
people_entity <- service$path("People")

## -----
## Method `ODataQuery$get`
## -----

service <- ODataQuery$new("https://services.odata.org/V4/TripPinServiceRW")
people_entity <- service$path("People")
russellwhyte <- people_entity$get("russellwhyte")

## -----
## Method `ODataQuery$func`
## -----

service <- ODataQuery$new("https://services.odata.org/V4/TripPinServiceRW")
get_nearest_airport <- service$func('GetNearestAirport')
## Not run:
get_nearest_airport(lat = 33, lon = -118)

## End(Not run)

## -----
## Method `ODataQuery$query`
## -----

service <- ODataQuery$new("https://services.odata.org/V4/TripPinServiceRW")
people_entity <- service$path("People")
people_entity$query(filter = "FirstName eq 'scott')$url

## -----
## Method `ODataQuery$top`
## -----

service <- ODataQuery$new("https://services.odata.org/V4/TripPinServiceRW")
people_entity <- service$path("People")
people_entity$top(10)

## -----
## Method `ODataQuery$skip`
## -----

service <- ODataQuery$new("https://services.odata.org/V4/TripPinServiceRW")

```

```

people_entity <- service$path("People")
people_entity$skip(10)

## -----
## Method `ODataQuery$select`
## -----

service <- ODataQuery$new("https://services.odata.org/V4/TripPinServiceRW")
people_entity <- service$path("People")
people_entity$select("FirstName", "LastName")

## -----
## Method `ODataQuery$filter`
## -----

service <- ODataQuery$new("https://services.odata.org/V4/TripPinServiceRW")
people_entity <- service$path("People")
people_entity$filter(FirstName.eq = 'Scott')

## -----
## Method `ODataQuery$expand`
## -----

service <- ODataQuery$new("https://services.odata.org/V4/TripPinServiceRW")
people_entity <- service$path("People")
people_entity$expand("Friends")

## -----
## Method `ODataQuery$orderby`
## -----

service <- ODataQuery$new("https://services.odata.org/V4/TripPinServiceRW")
people_entity <- service$path("People")
people_entity$orderby('Concurrency')
people_entity$orderby('-Concurrency')

## -----
## Method `ODataQuery$search`
## -----

service <- ODataQuery$new("https://services.odata.org/V4/TripPinServiceRW")
people_entity <- service$path("People")
people_entity$search('Boise')

## -----
## Method `ODataQuery$compute`
## -----

# Not really supported by this particular service.
service <- ODataQuery$new("https://services.odata.org/V4/TripPinServiceRW")
people_entity <- service$path("People")
people_entity$compute(a = "5 MUL Concurrency")

```

```

## -----
## Method `ODataQuery$retrieve`
## -----

## Not run:
service <- ODataQuery$new("https://services.odata.org/V4/TripPinServiceRW")
people_entity$retrieve()

## End(Not run)

## -----
## Method `ODataQuery$all`
## -----

## Not run:
service <- ODataQuery$new("https://services.odata.org/V4/TripPinServiceRW")
people_entity$all()
people_entity$all(jsonlite_args = list(simplifyVector = False))

## End(Not run)

## -----
## Method `ODataQuery$one`
## -----

## Not run:
service <- ODataQuery$new("https://services.odata.org/V4/TripPinServiceRW")
people_entity$top(1)$one(default = NA)

## End(Not run)

```

---

odata_function	<i>Make an OData function available to R</i>
----------------	--

---

### Description

This turns an OData function into an R function. Parameters are serialized to json. Scalar arguments should be passed as atomic vectors. Array or object arguments should be passed as list.

### Usage

```
odata_function(url, metadata = c("none", "minimal", "all"), ...)
```

### Arguments

url	Which url to fetch data from
metadata	Whether and how metadata is included
...	Arguments passed on to <a href="#">retrieve_data</a>
httr_args	List of additional arguments passed on to httr::GET
jsonlite_args	List of additional arguments passed on to jsonlite::fromJSON

**Value**

An R function

**See Also**

Other retrieve: [retrieve\\_all\(\)](#), [retrieve\\_data\(\)](#), [retrieve\\_one\(\)](#)

---

retrieve_all	<i>Retrieve data. If data is paged, concatenate pages. Only return the value without metadata.</i>
--------------	--

---

**Description**

Retrieve data. If data is paged, concatenate pages. Only return the value without metadata.

**Usage**

```
retrieve_all(url, ...)
```

**Arguments**

url	Which url to fetch data from
...	Arguments passed on to <a href="#">retrieve_data</a>
metadata	Whether and how metadata is included
httr_args	List of additional arguments passed on to httr::GET
jsonlite_args	List of additional arguments passed on to jsonlite::fromJSON

**See Also**

Other retrieve: [odata\\_function\(\)](#), [retrieve\\_data\(\)](#), [retrieve\\_one\(\)](#)

**Examples**

```
## Not run:
url <- "https://services.odata.org/V4/TripPinServiceRW/People"
retrieve_all(url)

## End(Not run)
```

---

retrieve_data	<i>Retrieve data</i>
---------------	----------------------

---

## Description

Retrieve data

## Usage

```
retrieve_data(  
  url,  
  metadata = c("none", "minimal", "all"),  
  httr_args = list(),  
  jsonlite_args = list()  
)
```

## Arguments

url	Which url to fetch data from
metadata	Whether and how metadata is included
httr_args	List of additional arguments passed on to httr::GET
jsonlite_args	List of additional arguments passed on to jsonlite::fromJSON

## Value

Data including metadata

## See Also

Other retrieve: [odata\\_function\(\)](#), [retrieve\\_all\(\)](#), [retrieve\\_one\(\)](#)

## Examples

```
## Not run:  
url <- "https://services.odata.org/V4/TripPinServiceRW"  
retrieve_data(url)  
  
## End(Not run)
```

---

retrieve_one	<i>Retrieve single instance.</i>
--------------	----------------------------------

---

### Description

Retrieve single instance.

### Usage

```
retrieve_one(url, default = stop("value not found"), ...)
```

### Arguments

url	Which url to fetch data from
default	The default if nothing was found. If not specified, an error is thrown in this case.
...	Arguments passed on to <a href="#">retrieve_data</a>
metadata	Whether and how metadata is included
httr_args	List of additional arguments passed on to <code>httr::GET</code>
jsonlite_args	List of additional arguments passed on to <code>jsonlite::fromJSON</code>

### Value

Single value or default if none. If the result consists of multiple records, an error is thrown.

### See Also

Other retrieve: [odata\\_function\(\)](#), [retrieve\\_all\(\)](#), [retrieve\\_data\(\)](#)

### Examples

```
## Not run:
url <- "https://services.odata.org/V4/TripPinServiceRW/People?$top=1"
retrieve_one(url)

url <- "https://services.odata.org/V4/TripPinServiceRW/People('russellwhyte')"
retrieve_one(url)

## End(Not run)
```

---

to_odata	<i>Macro to convert R to OData syntax</i>
----------	---

---

**Description**

Macro to convert R to OData syntax

**Usage**

```
to_odata(expr)
to_odata_(expr)
```

**Arguments**

expr                    Expression to convert to OData

**Details**

to\_odata takes unquote R code and quotes its input. Use !! to unquote an argument. to\_odata\_ requires its argument to be quoted already.

Only a subset of R is supported.

\* arithmetic The operators +, -, \*, / and

\* strings (characters in R) toupper, tolower, startsWith, endsWith, nchar, paste, paste0, trimws

\* arrays (lists in R) list, append, length

\* Formulae become lambdas in OData (x ~ x\$Name == "John")

Every unknown function is passed as is. If the function name is surrounded by percent signs it's treated as an infix operator.

**Examples**

```
to_odata(Field == value)
address <- "Bakerstreet 4"
to_odata(!address %in% Adresses)
to_odata(Friends$any(f ~ f$FirstName == 'John'))
```

# Index

## \* **retrieve**

- odata\_function, [11](#)
- retrieve\_all, [12](#)
- retrieve\_data, [13](#)
- retrieve\_one, [14](#)

and\_query, [2](#)  
and\_query(), [8](#)

not\_query (and\_query), [2](#)

odata\_function, [11](#), [12–14](#)  
ODataQuery, [3](#)  
or\_query (and\_query), [2](#)

retrieve\_all, [12](#), [12](#), [13](#), [14](#)  
retrieve\_data, [11](#), [12](#), [13](#), [14](#)  
retrieve\_one, [12](#), [13](#), [14](#)

to\_odata, [15](#)  
to\_odata\_ (to\_odata), [15](#)