# _ Filtering, sorting and paging and other GET parameters

In general each GET request to the API may receive filter conditions to filter the result by the content of the available fields and paging parameters to limit the amount of the resulting data.

Filters are provided as encoded JSON, containing all the fields and conditions to be used for filtering.

## Example:

After authenticating, and receiving a session id, the following GET could be called to get all tasks with a name containing the string "TM2".

The following example is copied directly from the communication between the translate5 browser application and the API:

http://demo.translate5.net/editor/task?_dc=1622533000384&page=1&start=0&limit=20&filter=%5B%7B%22operator%22%3A%22like%22%2C%22value%22%3A%22TM2%22%2C%22property%22%3A%22taskName%22%7D%5D

```
curl 'http://translate5.localdev/editor/task?_dc=1622533000384&page=1&start=0&limit=20&filter=%5B%7B%22operator%22%3A%22like%22%2C%22value%22%3A%22TM2%22%2C%22property%22%3A%22taskName%22%7D%5D' \
  -H 'Accept: application/json' \
  -H 'Cookie: zfExtended=YOUR_SESSION_ID'
```

### Parameters explained

| Parameter | Value | Explanation |
|---|---|---|
| start | number | the starting offset of the result rows to be returned. start=10 would start returning the results from the 10th found entry. Can be omitted and defaults to 0 then. |
| limit | number | the maximum count of entries to be returned in one request, can be omitted to return all found data. |
| filter | a JSON structure, urlencoded as string | A JSON filter configuration as URL encoded string. Details below. |
| sort | a JSON structure, urlencoded as string | A JSON sort configuration as URL encoded string. Details below. |
| _dc | a numeric timestamp | This parameter is not needed in pure API usage, the translate5 frontend adds it by default to bypass a potential server side caching of the data. |
| page | the numeric page of the displayed data | This parameter is not needed in pure API usage, the translate5 frontend calculates it and adds it by default to the request. |

## Filters explained

Each field of a REST entity may be used for filtering. The filters are provided as a JSON array containing several filter objects. So the filter string used in the above example will evaluate to:

```
[{
  "operator":"like",
  "value":"TM2",
  "property":"taskName"
}]
```

The API accepts multiple filter rules, each concatenated with AND.

To get all tasks containing "TM2" in the taskName and "XYZ" in the taskNr just add an additional filter:

```
[{
  "operator":"like",
  "value":"TM2",
  "property":"taskName"
},{
  "operator":"like",
  "value":"XYZ",
  "property":"taskNr"
}]
```

### Filter object in detail

Each filter object must contain the following attributes:

- operator: the operator to be used for comparsion, available operators depends on the data type of the used field
- value: the value to be used for the filter. The datatype depends on the used property and operator
- property: the name of the data property to be filtered

### Available operators

In general the following operators are available:

| Operator | applicable type | Explanation |
|---|---|---|
| like | Textual data | Searches for the given value in the given property with wildcards. So the above value TM2 will search not only for the exact name TM2 but for all tasks contain TM2 in its name. |
| in | all data | The value in the JSON is given as array, containing a list of values where the property must match at least one. |
| notInList | all data | Same as the in filter, just negated. |
| eq | number | Returns all entries with a numeric property matching exactly the given value. |
| gt | number | the property must be greater (>) then the given value. |
| gteq | number | the property must be greater or equal (>=) then the given value. |
| lt | number | the property must be lesser (<) then the given value. |
| lteq | number | the property must be lesser or equal (>=) then the given value. |
| = | boolean | Applicable for yes/no true/false fields. Value is either 0 or 1. |

## Sorting explained

Sorting is used similar to filtering, just with a different paremeter "sort". It is configured also by a JSON structure:

```
[{
        "property":"taskGuid",
        "direction":"ASC"
}]
```

The JSON can contain one or more sort objects. Each object contains the property to be used for sorting and the direction. Either ASC or DESC.