Querying endpoints

This article has been generated from the online version of the documentation and might be out of date. Please, make sure to always refer to the online version of the documentation for the up-to-date information.

Auto-generated at July 16, 2024

Resource Query Language

The *CloudBlue Connect* public API supports the RQL query language to search, sort and paginate objects.

A RQL query is a Javascript object that is defined according to the following typescript syntax:

```
interface IRQLFilter {
    $eq?: string|number,
    $ne?: string|number,
    $not?: IRQLFilter,
    $gt?: number,
    $ge?: number,
    $lt?: number,
    $le?: number,
    $like?: string,
    $ilike?: string,
    $empty?: boolean,
    $null?: boolean,
    $in?: Array<number|string>,
    $out?: Array<number|string>,
    $range?: {
        min: number,
        max: number,
    },
}

interface IQueryFilters {
    $or?: Array<IQueryFilters>;
    $ordering?: Array<string>;
    limit?: number;
    offset?: number;
    [key: string]?: string|number|Array<string|number>|boolean|IRQLFilter;
}
```

Examples

Simple filter

```
const filter = {
  name: 'eugene',
  age: 13,
};
```

Filter with text matching

```
const filter = {
```

```
  name: {
    $like: 'vasya*',
    $ilike: '***New',
  },
};
```

Filter with list

```
const filter = {
  age: {
    $out: [1, 2],
  },
  num: {
    $in: [3, 4, 5],
  },
};
```

Filter with range

```
const filter = {
  age: {
    $range: {
      max: 5,
      min: 9,
    },
  },
};
```

Filter with relational

```
const filter = {
  name: {
    $eq: 'vasya',
  },
  age: {
    $gt: 1,
    $lt: 8,
  },
  num: {
    $lte: 9,
    $gte: 4,
  },
};
```

Filter with logical NOT

```
const filter = {
  name: {
```

```
    $not: [{
      $eq: 'vasya',
    }, {
      $eq: 'petya',
    }],
  },
  age: {
    $not: {
      $eq: 10,
      $in: [1, 2, 3],
    },
  },
};
```

Filter with logical OR

```
const filter = {
  // You can use $or inside field
  color: {
    $or: [
      // Inside { } may be some conditions and for all them is used logical operator AND
      { $eq: 'red' },
      { $eq: 'blue' },
      { $eq: 'yellow' },
    ],
  },

  // Also you can use $or in root level, then inside must be objects array with fields name
  $or: [
    // Inside { } may be some fields with conditions and for all them is used logical operator AND
    { product: 'TV' },
    { product: 'Computer' },
  ],
};
```

Combine AND and OR filter

```
const filter = {
  $and: [
    {$or: [
      {status: 'new'},
      {type: 'program'}
    ]},
    {$or: [
      {status: 'done'},
      {type: 'service'}
    ]},
```

```
  ]
};
```

Filter with control operators

```
const controlFilter = {
  limit: 100,
  offset: 0,
  $ordering: '-created',
};
```

Filter with empty values

```
// If values are empty, null, undefined then they will not be in the query.
const filter = {
  $ordering: [],
  name: '',
  age: null,
  $or: [{name: undefined}],
  type: 'pending',
};
```

Combined

```
const combinationFilter = {
  offset: 0,
  limit: 10,
  $ordering: ['title', '-created'],
  $or: [
    {
      type: 'distribution',
      owner: { $eq: 'me' },
    },
    {
      type: { $in: ['sourcing', 'service'] },
      owner: { $not: { $eq: 'me' } },
    },
  ],
  name: {
    $or: [
      { $like: 'my test' },
      { $like: 'my' },
      { $ilike: '***CONTRACT' },
    ],
  },
};
```