

Filter

The filter parameter in the `loadrecords`

and

`loadobjecttypes`

functions is a complex JSON object representing the query criteria.

If the filter is omitted, no filter is applied.

A filter object can have different characteristics and also contain other filter objects. Multiple nested filters can therefore be defined.

Filter classes

Comparer

This filter performs one or more comparison operations on a field.

The properties of this filter have the name of a field and as value another object that describes the comparison operation and the comparison values:

```
{field name: {operator: value}}
```

Some comparison operations expect 2 or more comparison values. These are put into an array:

```
{field name: {operator: [value1, value2, ...]}}
```

Multiple comparison operations can also be specified for a single field by placing them in an array: (The comparison operations are ANDed, so they must all apply)

```
{field name: [{operator1: value1},{operator2: value2},...]}
```

Example:

```
{system_id: {$equals, "123456"}}  ≙  select * from tab where system_id='123456'
```

comparison operations

Operator	Value (data type)	Description	Example	corresponds to this SQL query
<code>\$isEmpty</code>	bool	true only; field has empty value; identical to <code>\$isNull</code>	<code>{system_id: {\$isEmpty: true}}</code>	<code>system_id IS NULL</code>
<code>\$notEmpty</code>	bool	true only; field has no empty value	<code>{system_id: {\$notEmpty: true}}</code>	<code>system_id <> ""</code>
<code>\$isNull</code>	bool	true only; field value is NULL	<code>{system_id: {\$isNull: true}}</code>	<code>system_id IS NULL</code>
<code>\$notNull</code>	bool	true only; field value is not NULL	<code>{system_id: {\$notNull: true}}</code>	<code>system_id IS NOT NULL</code>
<code>\$equals</code>	any	field value equals comparison value	<code>{system_id: {\$equals: "7"}}</code> shorthand: <code>{system_id: "7"}</code>	<code>system_id='7'</code>

\$notEquals	any	field value not equal to comparison value	{system_id: {\$notEquals: "7"}}	system_id <> '7'
\$endswith	string	field value ends with comparison value	{system_id: {\$endswith: "7"}}	system_id LIKE '%7'
\$startswith	string	field value starts with comparison value	{system_id: {\$startswith: "7"}}	system_id LIKE '7%'
\$contains	string	field value contains comparison value	{system_id: {\$contains: "7"}}	system_id LIKE '%7%'
\$like	string	field value matches the pattern	{system_id: {\$like: "%7_7%"}}	system_id LIKE '%7_7%'
\$in	any[]	field value equals one of the comparison values	{system_id: {\$in: ["7","8","9"]}}	system_id IN ('7','8','9')
\$notIn	any[]	field value not equal to any of the comparison values	{system_id: {\$notIn: ["7","8","9"]}}	system_id NOT IN ('7','8','9')
\$between	any[2]	field value lies between the two comparison values	{system_id: {\$between: ["7","9"]}}	system_id BETWEEN '7' AND '9'
\$notBetween	any[2]	field value not lies between the two comparison values	{system_id: {\$notBetween: ["7","9"]}}	system_id NOT BETWEEN '7' AND '9'
\$gt	any	field value is greater than comparison value	{system_id: {\$gt: "7"}}	system_id > '7'
\$gte	any	field value is greater than or equal to comparison value	{system_id: {\$gte: "7"}}	system_id >= '7'
\$lt	any	field value is less than comparison value	{system_id: {\$lt: "7"}}	system_id < '7'
\$lte	any	field value is less than or equal to comparison value	{system_id: {\$lte: "7"}}	system_id <= '7'
\$equals_field	string	field value is equal to specified other field	{system_last_change_at: {\$equals_field: "system_created_at"}}	system_last_change_at = system_created_at
\$notEquals_field	string	field value not equal to specified other field	{system_last_change_at: {\$notEquals_field: "system_created_at"}}	system_last_change_at <> system_created_at
\$gt_field	string	field value is greater than specified other field	{system_last_change_at: {\$gt_field: "system_created_at"}}	system_last_change_at > system_created_at
\$gte_field	string	field value is greater than or equal to specified other field	{system_last_change_at: {\$gte_field: "system_created_at"}}	system_last_change_at >= system_created_at
\$lt_field	string	field value is less than specified other field	{system_last_change_at: {\$lt_field: "system_created_at"}}	system_last_change_at < system_created_at

\$lte_field	string	field value is less than or equal to specified other field	{system_last_change_at: {\$lte_field: "system_created_at"}}	system_last_change_at <= system_created_at
\$between_field	string[2]	field value lies between the values of two other specified fields	{system_last_change_at: {\$between_field: ["system_created_at", "system_delete_at"]}}	system_last_change_at BETWEEN system_created_at AND system_delete_at
\$notBetween_field	string[2]	field value not lies between the values of two other specified fields	{system_last_change_at: {\$notBetween_field: ["system_created_at", "system_delete_at"]}}	system_last_change_at NOT BETWEEN system_created_at AND system_delete_at
\$date	int	Date/Time only: day of month equals comparison value	{system_created_at: {\$date: 1}}	DAY(system_created_at) = 1
\$day	int	Date/Time only: weekday equals comparison value	{system_created_at: {\$day: 1}}	DATEPART(dw, system_created_at) = 1
\$year	int	Date/Time only: year equals comparison value	{system_created_at: {\$year: 2000}}	YEAR(system_created_at)=2000
\$month	int	Date/Time only: month equals comparison value	{system_created_at: {\$month: 12}}	MONTH(system_created_at)=12
\$minutes	int	Date/Time only: minute equals comparison value	{system_created_at: {\$minutes: 59}}	DATEPART(mi, system_created_at) = 59
\$hours	int	Date/Time only: hour equals comparison value	{system_created_at: {\$hours: 18}}	DATEPART(hh, system_created_at) = 18

ALL

Default filter. This filter loads all records without restrictions.

Example:

```
{ $all: true } ? select * from tab where 1=1
```

only true is allowed

EMPTY

This filter creates an empty query. Usually not required.

Example:

```
{ $empty: true } ? select * from tab where 1=0
```

only true is allowed

NOT

This filter negates the included filter

Example:

```
{ $not: { system_id: "7" } } ? select * from tab where NOT (sytem_id='7')
```

AND

This filter connects the contained filters to a logical conjunction (AND)

Example:

```
{ $and: [{ vertragsnummer: { $contains: {"123"} }, { vertragsnummer: { $notEquals: "100123100" } }, ... ] } ? select * from tab where vertragsnummer like '%123%' AND vertragsnummer <> '100123100' AND ...
```

shorthand: { vertragsnummer: { \$contains: {"123"} }, vertragsnummer: { \$notEquals: "100123100" }, ... }

OR

This filter connects the contained filters to a logical disjunction (OR)

Example:

```
{ $or: [{ vertragsnummer: { $contains: {"123"} }, { vertragsnummer: { $notEquals: "100123100" } }, ... ] } ? select * from tab where vertragsnummer like '%123%' OR vertragsnummer <> '100123100' OR ...
```

Combination of Shorthands

There are shorthands (see above) for both the \$and filter and the \$equals comparator, which can be combined.

Example:

```
{ name: "Albrecht", vorname: "Klaus" } ? select * from tab where name='Albrecht' AND vorname='Klaus'
```

Nesting

\$and, \$or and \$not filters can be nested in any way.

Example:

```
{ $or: [{ $and: [filter1, filter2], $not: { $or: [filter3, filter4] } } ] } ? select * from tab where (filter1 AND filter2) OR NOT(filter3 OR filter4)
```

Erstellt: 2026-08-13 11:43:49 CEST von API

Zuletzt aktualisiert: 2026-08-13 11:43:49 CEST von API