iWD Manager Help

Search for tasks using iWD Query Language

4/6/2026

Contents

Search for tasks using iWD Query Language

Introduction

The iWD Query Language (QL) provides a flexible means for narrowing down the interactions presented in the Global Task List (GTL), with a number of operators including comparison and logical ones. It is a subset of Structured Query Language.

Constructing iWD QL queries

A simple query in QL (also known as a 'clause') consists of 3 parts, one following another: field, operator and value. Each of them is described on this page. Here's an example query:

```
Department = 'TEST'
```

This query will find all tasks in the TEST department. It uses the `Department` field, the `EQUALS` operator, and the value TEST.

Clauses can be combined by `AND` and `OR` operators and can be wrapped with round brackets. You can use brackets in complex iWD QL statements to enforce the precedence of operators.

For example, to find all tasks from either the `Finance` Department or with media type `email` and, from that selection, all tasks that have `Business Value` more than `100`, use the following query:

```
(Department = 'Finance' OR Media Type = 'e-mail') AND Business Value > 100
```

Fields

All tasks in the system have attributes and most of them can be used for filtering using iWD QL. The easiest way to find available attributes and their names is by accessing the **Build custom filter** menu by pressing the  button. You can also use the column names displayed in the GTL.

Operators

Task attributes contain data of specific types: `String`, `Number` or `Date`. Depending on the data type, only a specific set of operators is supported in query clauses, as shown below:

Type	Supported Operator
Number/Date	=, !=, >, >=, <, <=
String	=, !=, LIKE

EQUALS: =

The = operator is used to search for tasks where the value of the specified field exactly matches the value provided.

> **Important**
> String values should be wrapped in single quotes.

To find tasks where the value of a specified field exactly matches multiple values, use the IN LIST operator.

Example:

Find all tasks in the "TEST" department:

```
Department = 'TEST'
```

NOT EQUALS: !=

The != operator is used to search for tasks where the value of the specified field does not match the specified value.

Example:

Find all tasks that are not in the "TEST" department"

```
Department != 'TEST'
```

GREATER (LESS) THAN (OR EQUAL): >, >=, <, <=

The > (greater than) operator is used to search for tasks where the value of the specified field is greater than the specified value. The >= (greater than or equal to) does the same as > but also includes tasks whose value is equal to the one specified.

The < and <= operators ('less than' and 'less than or equal to' respectively) are the opposite of 'greater than' operators—they search for tasks whose value for the specified field is less than (or equal) to the specified value.

> **Important**

> Greater than and less than operators cannot be used with text values.

Examples:

Find all tasks where field `Priority` is more than 40 (exclusive) and less than 45 (inclusive): that is, tasks with priority being any of the following: 41, 42, 43, 44 and 45.

```
Priority > 40 AND Priority <= 45
```

Find all tasks where field `Business Value` more than or equal to 100 (inclusive) and less than 105 (exclusive): that is, tasks with Business Value being any of the following: 100, 101, 102, 103 and 104.

```
Business Value >= 100 AND Business Value < 105
```

LIKE: like

Determines whether a specific character string matches a specified pattern. A pattern can include regular characters and wildcard characters. During pattern matching, regular characters must exactly match the characters specified in the character string. However, wildcard characters can be matched with arbitrary fragments of the character string. Using wildcard characters makes the LIKE operator more flexible than using the = and != string comparison operators.

> **Important**
> The LIKE operator cannot be used with integer and date fields.

IN LIST: in

The `in` operator is used to search for tasks where the value of the specified field exactly matches one in the provided list.

> **Important**
> The IN operator cannot be used with date fields.

Examples

Find all tasks in departments TEST and OTHER_DEPARTMENT:

```
`Department in ('TEST', 'OTHER_DEPARTMENT')`
```

Find all tasks with priority either 100 or 200:

```
`Priority in (100, 200)`
```

Special characters

Wildcard characters

Character	Description
%	Any string with zero or more characters
_	Any single character search

Examples

Find all tasks in "TEST" Department:

```
`Department like 'TEST'`
```

Find all tasks where Department starts with "TEST":

```
`Department like 'TEST%'`
```

Find all tasks where Department ends with "TEST":

```
`Department like '%TEST'`
```

Find all tasks where Department contains "TEST":

```
`Department like '%TEST%'`
```

Find all tasks where Department starting with T, ending with 'T' and contains two any characters between them:

```
`Department like 'T__T'`
```

Reserved characters

The following characters are forbidden in string values: |, \, ?,], }, {, [, ", `.

> **Important**
>
> You can still use the \ symbol in order to escape wildcard characters % and _ to use them literally.

Here's an example of valid backslash use:

```
Department like '%TEST\%20'
```

This query will find all tasks where the Department ends with TEST%20. The same goes for the _ symbol.

If tasks contain any forbidden characters, you can use the single character wildcard '_' to omit their

explicit presence in the iWD QL query.

Working with dates

Absolute dates

iWD QL allows searching by date. You can search for tasks that were created on, before, or after a particular date (or date range). The date value must be written in the following format: 'YYYY-MM-DD', where YYYY is a full year, MM is a month with leading zero, and DD is a date with leading zero. Wrapping date values in quotation marks is not needed.

Examples:

Find all tasks where Task Due D/T is 31 Dec 2019:

```
Task Due D/T = 2019-12-31
```

Find all tasks where Task Due D/T is 31 Dec 2019 OR ask Due D/T is 1 Jan 2019:

```
Task Due D/T = 2019-12-31 OR Task Due D/T = 2019-01-01
```

Find all tasks where Task Due D/T between two dates 31 Dec 2019 and 1 Jan 2019:

```
Task Due D/T >= 2019-01-01 AND Task Due D/T =< 2019-12-31
```

Relative dates

You can specify a date relative to the current time. For example, you want to find all tasks with "Task Due D/T" in next 7 days:

```
Task Due D/T > 0d AND Task Due D/T < 7d
```

Assuming that today is 2019-12-12-23 16:23:34, the query above will find all tasks between now and 2019-12-21-30 16:23:34.

The following example will find all tasks with "Task Due D/T" in the previous 7 days:

```
Task Due D/T < 0d AND Task Due D/T > -7d
```

Supported relative values

Letter	Description	Example
y	year	Date > 1y
m	month	Date > 1m
d	day	Date > 1d
h	hour	Date > 1h

Relative values can be combined with each other. For example:

```
Task Due D/T > -3h8d2m1y
```

Values can be placed in any order. For example, "3h1m" does the same as "1m3h".

A zero value will do nothing. For example:

```
Task Due D/T > 0d
```

will find all tasks where "Task Due D/T" is past the current time.

Date shortcuts

iWD QL offers several date shortcuts (see table below) to use for quicker task filtering.

Shortcut	Applied value (assumed today = 2020-03-31)	Description
Today	2020-03-31	Today's date
Yesterday	2020-03-30	Today's date - 1 day
Tomorrow	2020-04-01	Today's date + 1 day
Next 7 days	7d	If applied with '>', finds all tasks later than 2020 April 6. If applied with '<', finds all tasks earlier than 2020 April 6.
Last 7 days	-7d	If applied with '>', finds all tasks later than 2020 March 24. If applied with '<', finds all tasks earlier than 2020 March 24.
This month	1m	If applied with '>', finds all tasks later than 2020 April 30. If applied with '<', finds all tasks earlier than 2020 April 30.
Last month	-1m	If applied with '>', finds all tasks later than 2020 February 29. If applied with '<', finds all tasks earlier than 2020 February 29.

> **Important**
> If the day of the month on the original date is greater than the number of days in the final month, the day of the month will change to the last day in the final month.

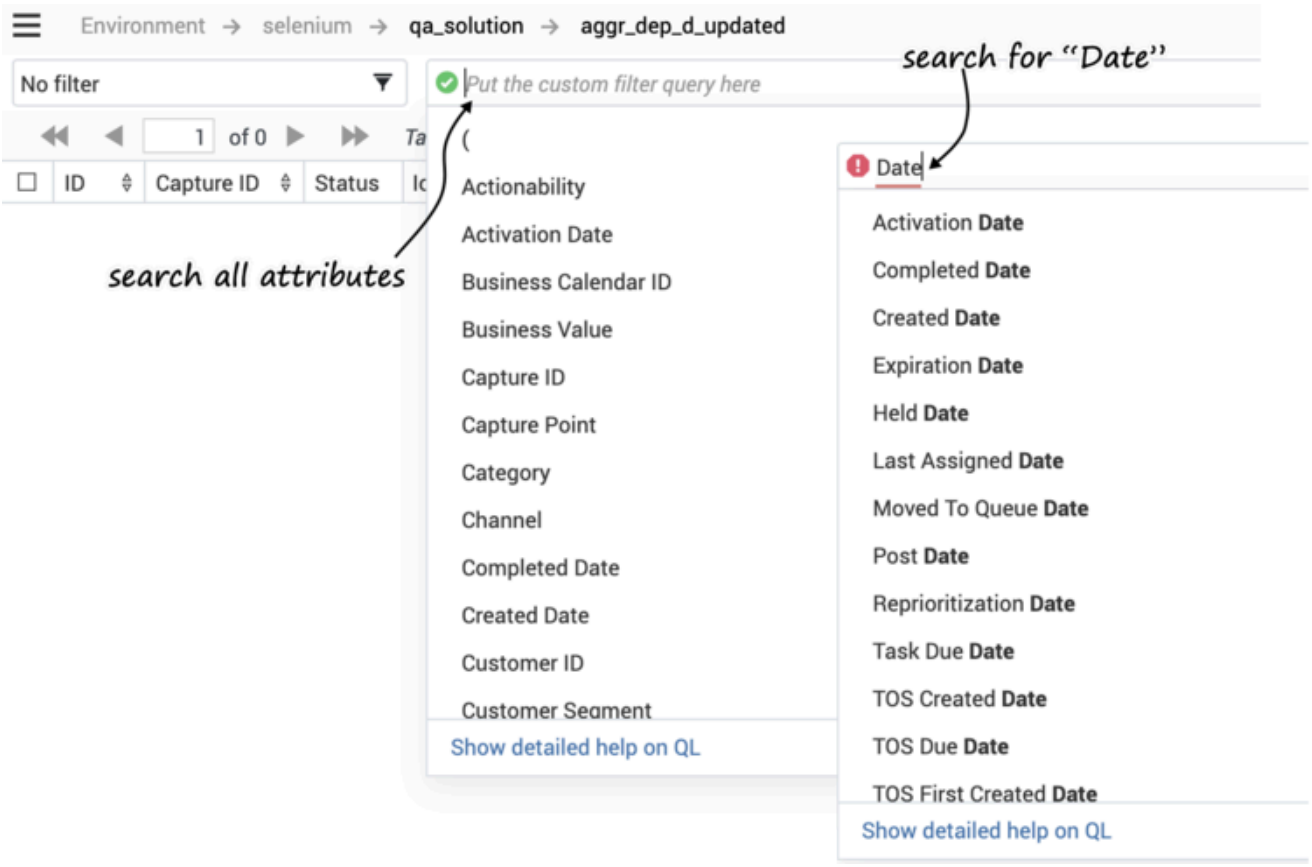
Autosuggest

The iWD Query Language provides suggestions to complete user inputs in several clicks for easier and faster use. This feature checks which entity should be added next and shows available variants in the list. Click on one of the variants to add it to the query.

The autosuggest mechanism works in cases of error or empty input and allows you to make corrections in a single click. It also supports keyboard navigation by using Up and Down arrows in the suggestion list. Press on Enter to apply a selected value. Click on the outside of the Autosuggest panel to hide it.

Fields

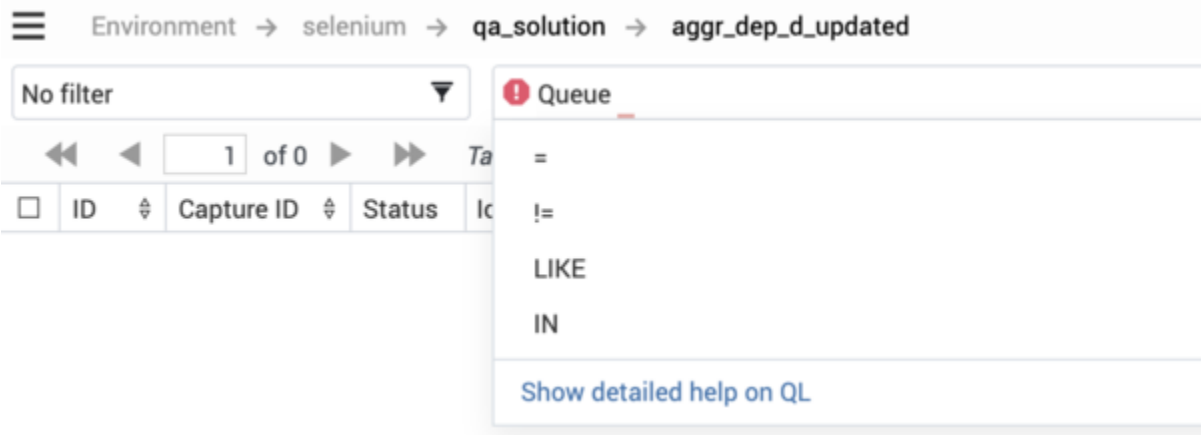
Attributes available for filtering are suggested at the start or after AND and OR operators. To reduce the number of suggestions, type a few characters from the desired attribute. Click on a suggested attribute to place it into iWD QL Input.



Operators

The autosuggest mechanism provides suggestions depending on which operators are supported by the attribute you enter. Click on a suggested operator to place it into iWD QL Input. For example, the

LIKE operator will not be suggested for Integer and Date attributes while the **IN LIST** operator will not be suggested for Date attributes.

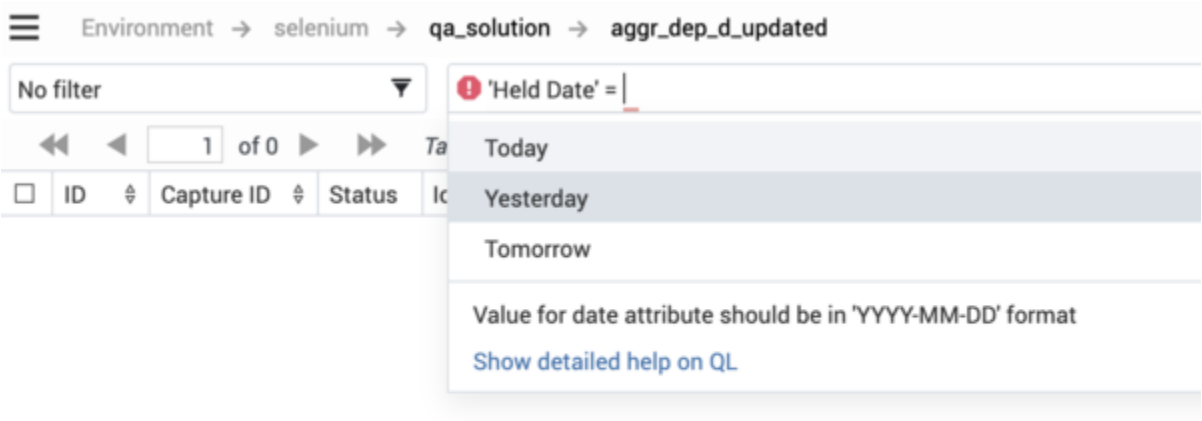


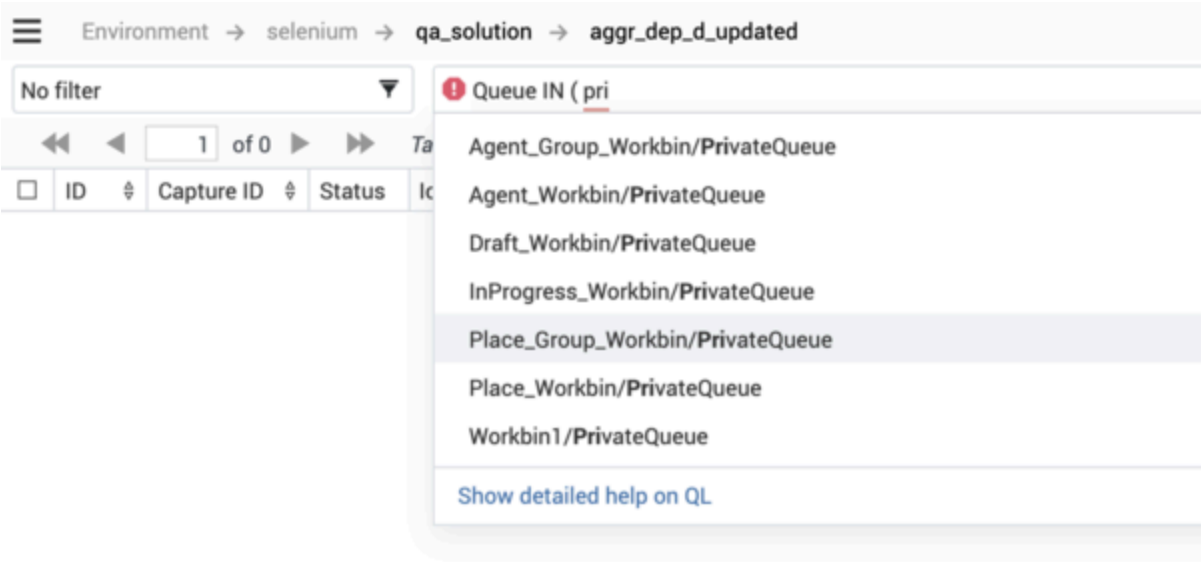
Values

With string attributes, the autosuggest mechanism checks whether the value you enter is wrapped in quote marks and suggests such wrapping in one click. Where an attribute supports a limited set of values, these values will be suggested with any operator.

With date attributes, the autosuggest mechanism offers several shortcuts: for more information see the Date shortcuts section.

With the **IN LIST** operator, the autosuggest mechanism will provide ',' and ')' as syntax symbols to continue and complete the clause respectively.





AND and OR

These operators will be suggested at the end of a clause to facilitate quicker typing.

Other suggestions

The autosuggest mechanism also suggests opening and closing brackets around the clauses.