

TicketQuery Wiki Macro

The TicketQuery macro lets you display ticket information anywhere that accepts [WikiFormatting](#). The query language used by the `[[TicketQuery]]` macro is described in the [TracQuery](#) page.

Usage

`[[TicketQuery]]`

Wiki macro listing tickets that match certain criteria.

This macro accepts a comma-separated list of keyed parameters, in the form "key=value".

If the key is the name of a field, the value must use the syntax of a filter specifier as defined in [TracQuery#QueryLanguage](#). Note that this is *not* the same as the simplified URL syntax used for `query:` links starting with a `?` character. Commas (,) can be included in field values by escaping them with a backslash (\).

Groups of field constraints to be OR-ed together can be separated by a literal `or` argument.

In addition to filters, several other named parameters can be used to control how the results are presented. All of them are optional.

The `format` parameter determines how the list of tickets is presented:

- **list** -- the default presentation is to list the ticket ID next to the summary, with each ticket on a separate line.
- **compact** -- the tickets are presented as a comma-separated list of ticket IDs.
- **count** -- only the count of matching tickets is displayed
- **rawcount** -- only the count of matching tickets is displayed, not even with a link to the corresponding query (*since 1.1.1*)
- **table** -- a view similar to the custom query view (but without the controls)
- **progress** -- a view similar to the milestone progress bars

The `max` parameter can be used to limit the number of tickets shown (defaults to **0**, i.e. no maximum).

The `order` parameter sets the field used for ordering tickets (defaults to **id**).

The `desc` parameter indicates whether the order of the tickets should be reversed (defaults to **false**).

The `group` parameter sets the field used for grouping tickets (defaults to not being set).

The `groupdesc` parameter indicates whether the natural display order of the groups should be reversed (defaults to **false**).

The `verbose` parameter can be set to a true value in order to get the description for the listed tickets. For **table** format only. *deprecated in favor of the `rows` parameter*

The `rows` parameter can be used to specify which field(s) should be viewed as a row, e.g.
`rows=description|summary`

The `col` parameter can be used to specify which fields should be viewed as columns. For **table** format only.

For compatibility with Trac 0.10, if there's a last positional parameter given to the macro, it will be used to specify the `format`. Also, using "&" as a field separator still works (except for `order`) but is deprecated.

Examples

Example	Result	Macro
Number of <u>Triage tickets</u> :	<u>0</u>	<code>[[TicketQuery(status=new&milestone=, count)]]</code>
Number of new tickets:	<u>0</u>	<code>[[TicketQuery(status=new, count)]]</code>
Number of reopened tickets:	<u>0</u>	<code>[[TicketQuery(status=reopened, count)]]</code>
Number of assigned tickets:	<u>0</u>	<code>[[TicketQuery(status=assigned, count)]]</code>
Number of invalid tickets:	<u>0</u>	<code>[[TicketQuery(status=closed, resolution=invalid, count)]]</code>
Number of worksforme tickets:	<u>0</u>	<code>[[TicketQuery(status=closed, resolution=worksforme, count)]]</code>
Number of duplicate tickets:	<u>0</u>	<code>[[TicketQuery(status=closed, resolution=duplicate, count)]]</code>
Number of wontfix tickets:	<u>0</u>	<code>[[TicketQuery(status=closed, resolution=wontfix, count)]]</code>
Number of fixed tickets:	<u>22</u>	<code>[[TicketQuery(status=closed, resolution=fixed, count)]]</code>
Number of untriaged tickets (milestone unset):	<u>0</u>	<code>[[TicketQuery(status!=closed, milestone=, count)]]</code>
Total number of tickets:	<u>22</u>	<code>[[TicketQuery(count)]]</code>
Number of tickets reported or owned by current user:	<u>0</u>	<code>[[TicketQuery(reporter=\$USER, or, owner=\$USER, count)]]</code>
Number of tickets created this month:	<u>0</u>	<code>[[TicketQuery(created=thismonth.., count)]]</code>
	<u>0</u>	<code>[[TicketQuery(status=closed, keywords~=firefox, count)]]</code>

`[[TicketQuery]]`

Example	Result	Macro
Number of closed Firefox tickets:		
Number of closed Opera tickets:	0	<code>[[TicketQuery(status=closed,keywords~=opera,count)]]</code>
Number of closed tickets affecting Firefox and Opera:	0	<code>[[TicketQuery(status=closed,keywords~=firefox opera,count)]]</code>
Number of closed tickets affecting Firefox or Opera:	0	<code>[[TicketQuery(status=closed,keywords~=firefox opera,count)]]</code>
Number of tickets that affect Firefox or are closed and affect Opera:	0	<code>[[TicketQuery(status=closed,keywords~=opera,or,keywords~=firefox,count)]]</code>
Number of closed Firefox tickets that don't affect Opera:	0	<code>[[TicketQuery(status=closed,keywords~=firefox -opera,count)]]</code>
Last 3 modified tickets:	#22, #21, #20	<code>[[TicketQuery(max=3,order=modified,desc=1,compact)]]</code> <code>[[TicketQuery(id=1,col=id owner reporter,rows=summary,table)]]</code>
Details of ticket #1:	<u>Ticket</u> #1	<u>Owner</u> kane
	<u>Reporter</u> kane	
	<u>Summary</u>	<u>Trace</u>

Format: list

```
[[TicketQuery(version=0.6|0.7&resolution=duplicate)]]
```

This is displayed as:

No results

```
[[TicketQuery(id=123)]]
```

Examples

This is displayed as:

No results

Format: compact

```
[[TicketQuery(version=0.6|0.7&resolution=duplicate, compact)]]
```

This is displayed as:

No results

Format: count

```
[[TicketQuery(version=0.6|0.7&resolution=duplicate, count)]]
```

This is displayed as:

0

Format: progress

```
[[TicketQuery(milestone=0.12.8&group=type, format=progress)]]
```

This is displayed as:

Format: table

You can choose the columns displayed in the table format (`format=table`) using `col=<field>`. You can specify multiple fields and the order they are displayed by placing pipes (|) between the columns:

```
[[TicketQuery(max=3, status=closed, order=id, desc=1, format=table, col=resolution|summary|owner|reporter)]]
```

This is displayed as:

Results (1 - 3 of 22)

1 2 3 4 5 6 7 8 9

<u>Ticket</u>	<u>Resolution</u>	<u>Summary</u>	<u>Owner</u>	<u>Reporter</u>
#22	fixed	<u>Component "Execute loop"</u>	somebody	rosiere
#21	fixed	<u>Component "Register unit"</u>	somebody	rosiere
#20	fixed	<u>Component "Status list"</u>	somebody	rosiere

1 2 3 4 5 6 7 8 9

Full rows

In *table* format you can specify full rows using `rows=<field>`:

```
[[TicketQuery(max=3, status=closed, order=id, desc=1, format=table, col=resolution|summary|owner|reporter, rows=id)]]
```

This is displayed as:

Format: list

Results (1 - 3 of 22)

1 2 3 4 5 6 7 8 ?

<u>Ticket</u>	<u>Resolution</u>	<u>Summary</u>	<u>Owner</u>	<u>Reporter</u>
<u>#22</u>	fixed	<u>Component "Execute loop"</u>	somebody	rosiere
Intégration des composants:				

Description

- Execute_unit
- Read_unit
- Register_unit
- Write_unit

<u>#21</u>	fixed	<u>Component "Register unit"</u>	somebody	rosiere
Intégration des composants :				

Description

- Register_file
- Status_list

<u>#20</u>	fixed	<u>Component "Status list"</u>	somebody	rosiere
Composant status list : contient une ram qui indique si un registre est valide ou non				

Description

- lors de l'allocation : mise à 0 de ce bit.
- lors de l'écriture : mise à 1 de ce bit.

1 2 3 4 5 6 7 8 ?

See also: [TracQuery](#), [TracTickets](#), [TracReports](#)