

ConfiForms Filters



Looking for [CLOUD](#) documentation? This page has information that is valid for both versions

Filters / Conditions



Filters is a **very important concept** in ConfiForms and is used in many places, basically it is the core of ConfiForms, a base which helps to connect ConfiForms forms and allows to query data stored within ConfiForms form:

- to limit number of records returned by "**views" macros (TableView, ListView, CardView, CalendarView and ValueView)
- to fire IFTTT actions based on conditions (ConfiForms IFTTT macro)
- to execute rules based on a condition (ConfiForms Field Definition Rules)
- to define custom CSS rules for your views, conditionally



See also [Accessing field values and properties](#). You can use complex properties in your filters. For example filtering dropdown fields by values and by labels, filtering page type fields by page metadata fields, filtering user fields by, for example - email property

Filter expressions can be used with [Virtual functions](#)



See [ConfiForms Filters by example](#) to learn by example

ConfiForms plugin uses 'Lucene like' syntax for expressions, supports grouping using parenthesis and AND/OR logical operators. Nesting expressions is possible

Supports filtering for expressions that start with a wild-card. Supports filtering per field as well as free text search (filter to match against any field value).

Operators AND and OR are **case-sensitive!** (as everything else in ConfiForms: field names, form names, virtual functions)

Supports not notation, to filter records that do not match the given filter: '!' is used for that. Example: !f1:[empty] - will look for records where f1 field value is not empty (has some value)

Supports simple math operations and comparisons on dates (to add or subtract days from today's date).

Valid example: (f1:*success AND f2:>[yesterday]) OR (f3:accepted) - matches records with field values having 'success' and where f2 field value is after yesterday or where f3 field value is equals to 'accepted'

Valid example: (f1:*success AND f2:>[today]-1) OR (!f3:[empty]) - matches records with field values having 'success' and where f2 field value is after yesterday or where f3 field value is not empty

Through "evaluateFormula" from [Virtual functions](#) you have access to [Supported math operators, formulas and functions](#)

Main principle is you match current record's field's value with a given value

```
fieldname:value
```

where fieldname itself could be an expression and value could be a reference to field values via [entry.field_name] notation

```
fieldname.trunc(3):[entry.anotherfield.trunc(3)]
```

Important to follow the concept of a **field name** followed by a **value** to check against separated with :

To match values against regular expressions please consider using "matches" function from [Virtual functions](#)

```
myfield.matches(^[a-zA-Z0-9]*$):true
```

Reserved words to use in expressions:

| | |
|---------------------------------|---|
| * | To match all |
| val* | Will match the record(s) where the field value starts with "val" |
| *val | Will match the record(s) where the field value ends with "val" |
| *val* | Will match the record(s) where the field value has "val" anywhere |
| fieldname:val* | Will only check the "fieldname" field for the value "val" in the beginning of each word in the value |
| [empty] | To match empty values for particular field. Example: field1:[empty] - will match records where field1 is empty (does not have a value, but the form defines this field) |
| [now] | Current time and date, useful with '<' and '>' for comparing with dates stored. Example: someDateField:<[now] - will match records where field 'someDateField' has value which is in the past compared to now (current time) |
| [today] | Same as [now], but without time |
| [tomorrow] | To compare against tomorrow's date. Also something like [today]+1 could be used instead |
| [yesterday] | To compare against yesterday's date. Also something like [today]-1 could be used instead |
| [dateyyyyMMdd] | To compare against given date in the format: yyyyMMdd, example: [date20150130] to give a date as Jan 30 2015 |
| [datetimeyyyyMMdd HH:mm] | To compare against given date with time in the format: yyyyMMdd HH:mm, example: [date20151231 12:13] to give a date as Dec 31 2015 12:13 |
| [date timetoday hh:mm] | You can use a shortcut to "today" with given construction |
| '<' and '>' (and '<=' and '>=') | Can be used together with date and datetime fields, as well as to compare values for numeric fields stored |
| ! | To reverse the filter condition. Example: !field1:[empty] - will find records that have 'field1' field filled |
| this | <p>In *views macros you can reference current user and current page as "this" (when used to filter user (and multiuser) and page (and autopage) field types respectfully).</p> <p>Example:</p> <ul style="list-style-type: none">thepage: thiscreatedBy: this <p>(thepage is the field of type Page and createdBy is a metadata field to hold created by info for given record). See more about metadata field and available field types in Documentation</p> |



From version **1.35** we start to deprecate support for ".this" in filters, as filters are now "context aware" and you can use

- `_user`
- `_today`
- `_page`

as any other ConfiForms field to get, current user (User object), date (DateHolder) and page (Page object) where the filter is executed

These "context fields" work with [Virtual functions](#)

```
--- format current date to day of week
_today.formatDate(u)

--- get current page ID
_page.id

--- some weird transformation example to get the page title and split it by
spaces then join by a single quote
_page.displayTitle.split( ).toArray(')

--- get current user's location (through profile)
_user.asUserProfile.location
```

Also, there is a "_count" field which holds the number of record in the current scope (might change in time while conditions get processed by a filter engine)

That said, consider the following example:

- You have 5 records initially and the filter looks like

```
_count:>4
```

will match all

- While the filter like this

```
somefield:hello AND _count:>4
```

will first try to find the matches for "hello" and then apply the other condition (and depending on matches on left-hand side of the query the count might be less than 4)



If you want to use the values of context field in your filters then you will need to use it through the `[entry.fieldname]` reference

Something like:

```
ownedBy:[entry._user]
```

```
mypage:[entry._page.id]
```


hasChanged(fieldName)

FROM V. 1.36

See explanation in [Virtual functions](#)



This function is supported only when filter is used in condition field in IFTTT macro, as only in this case there is an information about the previous state of the record present in the context

| | |
|--|--|
| _previousState.fieldname | <p>See hasChanged function in Virtual functions</p> <div>  This function is supported only when filter is used in condition field in IFTTT macro, as only in this case there is an information about the previous state of the record present in the context </div> <p>You can access ANY property of the ConfiForms record that was there before the update. But ONLY in the IFTTT conditions</p> |
| <p>Supports for "records count" evaluation</p> <p>FROM V. 1.52.6</p> | <p>For example, this filter will check that the user can register not more than 2 records within 14 days</p> <pre>(createdBy:[entry._user] AND created:>([today]-14)):>2</pre> <p>So, the filter shall start with a parenthesis and end with a parenthesis followed by semicolon and expression</p> <pre>> < >= <= or just a number (to match exactly)</pre> <p>This works in Field Definition Rules and allows you to have validation rules as dynamic as this</p> |
| <p>Calculations on date /datetime and timestamp fields</p> | <p>When you want to apply filters on date/datetime or timestamp fields you actually work on their "timestamp values" - https://www.unixtimestamp.com/index.php</p> <p>And that is in milliseconds. So, when you want to add/subtract days or hours or minutes from some timestamp holding field you need to operate in milliseconds</p> <p>For example when you want to check the interval is longer than 60 days (duration is a DateTime interval field in this example)</p> <pre>duration.startDate:>([entry.duration.endDate]-5184000000)</pre> <p>How we got a "5184000000" value? This is 60 days in milliseconds</p> <p>How to convert a day into milliseconds: https://www.unitjuggler.com/convert-time-from-day-to-ms.html</p> |

Some examples:

- *field1:[today]-5* - assuming field1 is of type date (or datetime) this filter will return records where field1 value is not older than 5 days from now
- *field1:[today]+10* - assuming field1 is of type date (or datetime) this filter will return records where field1 value is not after 10 days from now


Since ConfiForms version 1.13

We now support the following constants in field names:


| Field name constant | How to use | Details |
|------------------------------|---------------------------------|--|
| [count] _count | _count:<3 | Will check for records count in the resultset and will return an empty result if this condition is not met. Important: this works better when you add this condition as last one, meaning that other filters were already applied on the dataset and you need to check the count of that filtered result |
| [today] | _today:> [date201512 10] | For queries to run on a certain date (when date given is a constant). Will check today's date against the given date. In this example: the query will match when today's date is AFTER the 10 Dec 2015 |
| [now] _now | _now:<[date time201512 10 2:30] | When "now" (current time/date) is before the given date/time |

| | | |
|--------------------|--|---|
| [user] | <code>_user: someuserna me</code> | Brings the user context into query execution. Available from ConfiForms version 1.17+ |
| <code>_user</code> | <code>!_user: someuserna me</code> | Anonymous users are evaluated as "" |


All fields support >, <, <=, >= and =



Since version 1.24.x of ConfiForms we had to change the constants for "field names" in filters from ~~[count]~~ to `_count`. This only concerns the field names constants you can use in filters to access the values of the current context, such as, current time, user and records count. See the table above




Some field types (ConfiForms Field Definition), such as user types and page types could use "this" to reference to current user or current page (where the filter is used). See below



See also [Accessing field values and properties](#)

How to filter by:

| | |
|--|--|
| Show all records | * |
| Show records where text field called "mytext" starts with the value "this is test" | mytext:this is test* |
| Show records where text field called "mytext" has the value "this is test" | mytext:*this is test* |
| Show records where text field called "mytext" ends with the value "this is test" | mytext:*this is test |
| Records created today | created:[today] |
| | <div>  <p>It is better to operate with timestamps on date/date time fields, see more Accessing field values and properties</p> </div> |
| Records created before today | created:<[today] |
| Records created yesterday | created:<[today]-1 |
| Records created yesterday (alternative to previous example) | created:<[yesterday] |
| Records owned by current user | ownedBy:this Use instead: ownedBy:[entry._user] |
| Records created by current user | createdBy:this Use instead: createdBy:[entry._user] |
| When you have autopage type of field, that sets automatically the page where the record was created. You might want to filter by that, having a ListView, CardView or TableView on it. Let's say the autopage field is named "apage" | apage:this Use instead: apage.id:[entry._page.id] |

| | |
|--|--|
| How to filter, if field type is Page or Autopage: use Confluence pageld numeric value in filters | somepagefield:pageld |
| Assuming you have a field of type "Page" and the field is called "mypage". And you want to filter by page title. This is totally possible, as ConfiForms allows you to access rich properties easily. | mypage.title:Some title* See Accessing field values and properties for more details |



See [Accessing field values and properties](#) to understand which properties and fields a field has. Any accessible field could be used in a filter, both on the left side of the expression and on the right side of the expression

Examples:

| Left side | Separator | Right side |
|-------------------------------------|-----------|----------------------|
| myfield | : | [entry.field2.title] |
| somefield.id | : | 2 |
| mysmartfield.anothersmartfield.name | : | Alex* |