

Managing Free Text Filters

This version of the app's documentation is outdated. Please find the information you're looking for here:

- [Filter types available](#)

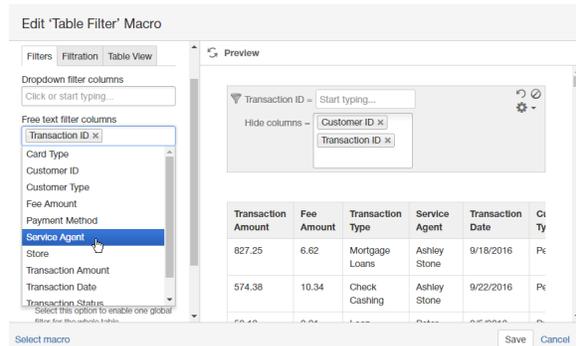
Managing Free Text Filters

Table Filter for Confluence add-on is equipped with the free text filters. You can add this filter type for any column of your table. It allows you to enter text queries for filtration of data in the selected column of your table. You can also enter the regular expressions for flexible data filtration in the column.

- [This version of the app's documentation is outdated. Please find the information you're looking for here:](#)
- [Adding the Free Text Filter](#)
- [Filtering the table with free text filters](#)
- [Using regular expressions](#)
 - [Quick Recipes with Regular Expressions](#)

Adding the Free Text Filter

1. Edit the page.
2. [Insert the Table Filter macro](#) and paste the table within the macro body.
3. Select the macro and click **Edit**.
4. In the **Free text filter columns** box, select the columns for filtration with free text filters.
5. Save the macro.
6. Save the page.



Filtering the table with free text filters

1. Open the page with the table for filtration.
2. On the filtration pane, locate the free text filters and position the mouse pointer within the appropriate one.
3. Enter the appropriate query or regular expression for filtration.



You can also filter tables with @-user mentions by the current user (write or select **@currentUser**).

Transaction ID	Transaction Amount	Fee Amount	Transaction Type	Service Agent	Transaction Date	Customer Type	Transaction Status	Customer ID	Card Type	Payment Method	Store
520498528	53.13	0.21	Loan Settlement	Peter Jacobs	9/5/2016	Business	Processed	1796B0DQA3412	Debit	job	104
541296507	337	3.37	Loan Settlement	Peter Jacobs	9/7/2016	Mixed	Rejected	2313DRHC1520	Credit	mastercard	106
505986162	891.13	1.76	Mortgage Loans	Jack Hill	9/29/2016	Business	Archived	1146CJUEA1083	Debit	bankcard	103
562888027	461.63	3.69	Mortgage Loans	Jack Hill	9/9/2016	Mixed	Pending	3411DKGUA1711	Debit	job	104
571897241	273.75	0.55	Mortgage Loans	Tom Oliver	9/24/2016	Mixed	Processed	1303FCHDR0209	Credit	visa-electron	108
517188341	748.75	4.49	Mortgage Loans	Jack Hill	9/4/2016	Personal	Pending	1404AFFB83604	Credit	bankcard	102
547707459	59.5	0.36	Mortgage Loans	Jack Hill	9/27/2016	Personal	Pending	2733JUDG1402	Credit	job	106

Using regular expressions

You can use JavaScript-style regular expressions in free text filters.

Regular Expression	Matched Values
[Dd]oe	Doe, doe
colo(u)?r	color, colour
Developer Scientist	Developer, Scientist

Table Filter macro uses the **OR** operator for table filtration. You can use the **AND** operator in the free text and global filters. Enter the **&** ('ampersand') between values to match two or more values in the cell at once.

The full list of regular expressions is available [here](#).

Free text filters allow you to use regular expressions while filtering table data. They allow you to write powerful filtration queries.

Quick Recipes with Regular Expressions

Regular Expression	Explanation	Example														
<code>\d\d</code>	This query will look for two digits separated by the point (for example, '8.1', '9.5'). Entries with more than one digit after the point will be also included.	<p>If applying the <code>\d\d</code> query to the Operating System column, you will get three values with 'OSx 5.7', 'cOS 7.2' and 'aOS 2.8' after filtration.</p> <table border="1"> <thead> <tr> <th>User</th> <th>Operating System</th> </tr> </thead> <tbody> <tr> <td>Mike</td> <td>OSx 5.7</td> </tr> <tr> <td>Jane</td> <td>SuperOS</td> </tr> <tr> <td>Stewart</td> <td>OSx 9</td> </tr> <tr> <td>Jane</td> <td>cOS 7.2</td> </tr> <tr> <td>Andy</td> <td>aOS 2.8</td> </tr> </tbody> </table>	User	Operating System	Mike	OSx 5.7	Jane	SuperOS	Stewart	OSx 9	Jane	cOS 7.2	Andy	aOS 2.8		
User	Operating System															
Mike	OSx 5.7															
Jane	SuperOS															
Stewart	OSx 9															
Jane	cOS 7.2															
Andy	aOS 2.8															
<code>\d\d:\d\d:</code>	You can use this query if you want to filter dates having time and dates without time.	<p>If applying the <code>\d\d:\d\d:</code> query to the Sync Time column, you will get three values with '11/17/2015 10:35:58', '11/17/2015 12:24:54' and '11/19/2015 16:47:22' after filtration.</p> <table border="1"> <thead> <tr> <th>User</th> <th>Sync Time</th> </tr> </thead> <tbody> <tr> <td>Mike</td> <td>11/17/2015 10:35:58</td> </tr> <tr> <td>Jane</td> <td>11/28/2015</td> </tr> <tr> <td>Stewart</td> <td>11/17/2015 12:24:54</td> </tr> <tr> <td>Jane</td> <td>11/30/2015</td> </tr> <tr> <td>Molly</td> <td>11/16/2015 17:28</td> </tr> <tr> <td>Andy</td> <td>11/19/2015 16:47:22</td> </tr> </tbody> </table>	User	Sync Time	Mike	11/17/2015 10:35:58	Jane	11/28/2015	Stewart	11/17/2015 12:24:54	Jane	11/30/2015	Molly	11/16/2015 17:28	Andy	11/19/2015 16:47:22
User	Sync Time															
Mike	11/17/2015 10:35:58															
Jane	11/28/2015															
Stewart	11/17/2015 12:24:54															
Jane	11/30/2015															
Molly	11/16/2015 17:28															
Andy	11/19/2015 16:47:22															
<code>^[0-9]{2}\$</code> <code>^[0-9]{3}\$</code>	You can use these queries if you want to filter people by age depending on the number of digits.	<p>If applying the <code>^[0-9]{2}\$</code> query to the Age column, you will get three numbers comprised of two digits.</p> <table border="1"> <thead> <tr> <th>User</th> <th>Age</th> </tr> </thead> <tbody> <tr> <td>Mike</td> <td>120</td> </tr> <tr> <td>Jane</td> <td>75</td> </tr> <tr> <td>Stewart</td> <td>57</td> </tr> <tr> <td>Jane</td> <td>111</td> </tr> <tr> <td>Molly</td> <td>85</td> </tr> </tbody> </table>	User	Age	Mike	120	Jane	75	Stewart	57	Jane	111	Molly	85		
User	Age															
Mike	120															
Jane	75															
Stewart	57															
Jane	111															
Molly	85															
<code>\b([1-1][1-1])</code> <code>\b([1-1][2-2])</code> <code>\b([1-1][3-3])</code> and so on	You can use this query if you have a list with timestamps and you would like to see how many operations were performed at a particular hour,	<p>If applying the <code>\b([1-1][1-1])</code> query to the Login time column, you will get three entries of login attempts.</p> <table border="1"> <thead> <tr> <th>User</th> <th>Login Time</th> </tr> </thead> <tbody> <tr> <td>Mike</td> <td>11:30:25</td> </tr> <tr> <td>Jane</td> <td>13:31:25</td> </tr> <tr> <td>Stewart</td> <td>11:32:25</td> </tr> <tr> <td>Jane</td> <td>12:40:25</td> </tr> <tr> <td>Molly</td> <td>11:47:25</td> </tr> <tr> <td>Stewart</td> <td>12:20:25</td> </tr> </tbody> </table>	User	Login Time	Mike	11:30:25	Jane	13:31:25	Stewart	11:32:25	Jane	12:40:25	Molly	11:47:25	Stewart	12:20:25
User	Login Time															
Mike	11:30:25															
Jane	13:31:25															
Stewart	11:32:25															
Jane	12:40:25															
Molly	11:47:25															
Stewart	12:20:25															

<p>h(n a)s</p>	<p>You can use this query if you want to filter last names having either 'n' or 'a' between h and s.</p>	<p>If applying the 'h(n a)s' query to the Last Name column, you will get four entries (Johns, Johnston, Johnson and Johason).</p> <table border="1" data-bbox="868 205 1156 619"> <thead> <tr> <th>First Name</th> <th>Last Name</th> </tr> </thead> <tbody> <tr> <td>Alexander</td> <td>Johns</td> </tr> <tr> <td>John</td> <td>Johnston</td> </tr> <tr> <td>Mike</td> <td>Johney</td> </tr> <tr> <td>Alice</td> <td>Johnson</td> </tr> <tr> <td>Pavel</td> <td>Johason</td> </tr> <tr> <td>Jane</td> <td>Johannesen</td> </tr> <tr> <td>Jannet</td> <td>Johndrow</td> </tr> <tr> <td>Paul</td> <td>Johnting</td> </tr> </tbody> </table>	First Name	Last Name	Alexander	Johns	John	Johnston	Mike	Johney	Alice	Johnson	Pavel	Johason	Jane	Johannesen	Jannet	Johndrow	Paul	Johnting
First Name	Last Name																			
Alexander	Johns																			
John	Johnston																			
Mike	Johney																			
Alice	Johnson																			
Pavel	Johason																			
Jane	Johannesen																			
Jannet	Johndrow																			
Paul	Johnting																			
<p>Ruby & Python</p>	<p>You can use '&' (ampersand) to filter cells consisting two or more values standing in different positions in the cells.</p> <div data-bbox="310 709 846 821" style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"> <p> This is a custom enhancement in the add-on that simplifies the usage of the regular expression to filter cells containing two or more values at a time.</p> </div>	<p>If applying the 'Ruby & Python' query to the Skills column, you will get two names of employees who have these skills.</p> <table border="1" data-bbox="868 709 1385 1031"> <thead> <tr> <th>User</th> <th>Skills</th> </tr> </thead> <tbody> <tr> <td>Mike</td> <td>Java, JavaScript, HTML5, CSS, ASP, Agile</td> </tr> <tr> <td>Jane</td> <td>Java, PHP, CSS, HTML5, Agile, Ruby</td> </tr> <tr> <td>Stewart</td> <td>SOAP, JSON, PHP, CSS, Java, REST, C++</td> </tr> <tr> <td>Jane</td> <td>Java, Ruby, Python, PHP, CSS, HTML5</td> </tr> <tr> <td>Molly</td> <td>PHP, CSS, Java, JavaScript, HTML5</td> </tr> <tr> <td>Stewart</td> <td>Ruby, HTML5, PHP, CSS, JavaScript, Python</td> </tr> </tbody> </table>	User	Skills	Mike	Java, JavaScript, HTML5, CSS, ASP, Agile	Jane	Java, PHP, CSS, HTML5, Agile, Ruby	Stewart	SOAP, JSON, PHP, CSS, Java, REST, C++	Jane	Java, Ruby, Python, PHP, CSS, HTML5	Molly	PHP, CSS, Java, JavaScript, HTML5	Stewart	Ruby, HTML5, PHP, CSS, JavaScript, Python				
User	Skills																			
Mike	Java, JavaScript, HTML5, CSS, ASP, Agile																			
Jane	Java, PHP, CSS, HTML5, Agile, Ruby																			
Stewart	SOAP, JSON, PHP, CSS, Java, REST, C++																			
Jane	Java, Ruby, Python, PHP, CSS, HTML5																			
Molly	PHP, CSS, Java, JavaScript, HTML5																			
Stewart	Ruby, HTML5, PHP, CSS, JavaScript, Python																			