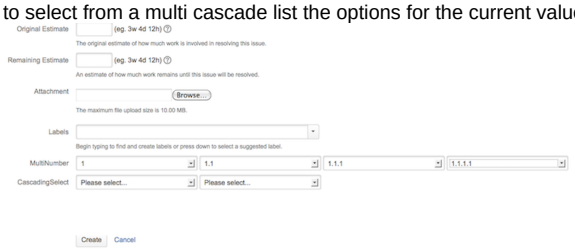
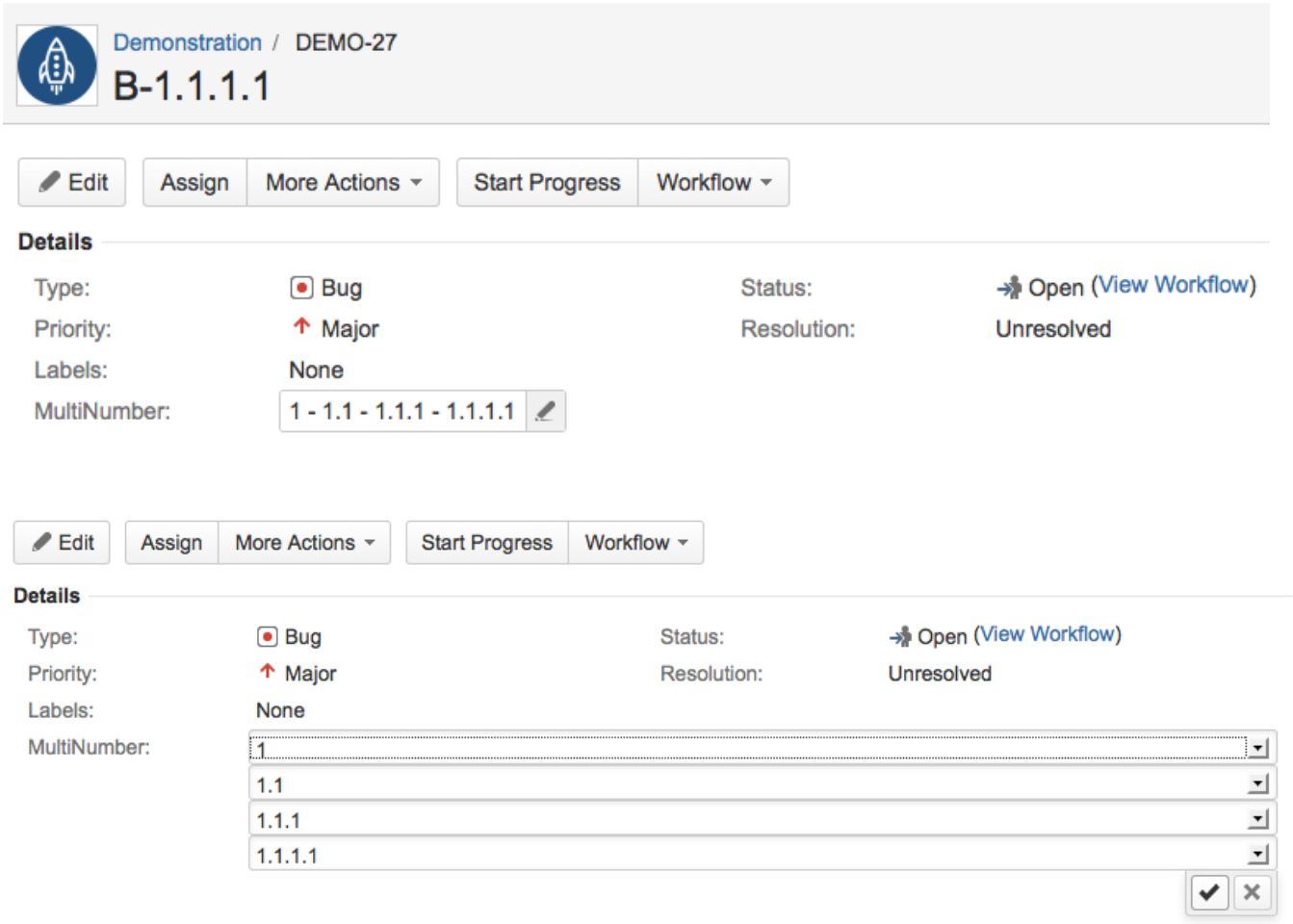
User Guide (Server / DC)

Issue creation and edit

Creating and editing an issue with a Multi level Cascade custom field value is really a simple task. During the creation of an issue is possible to select from a multi cascade list the options for the current value.

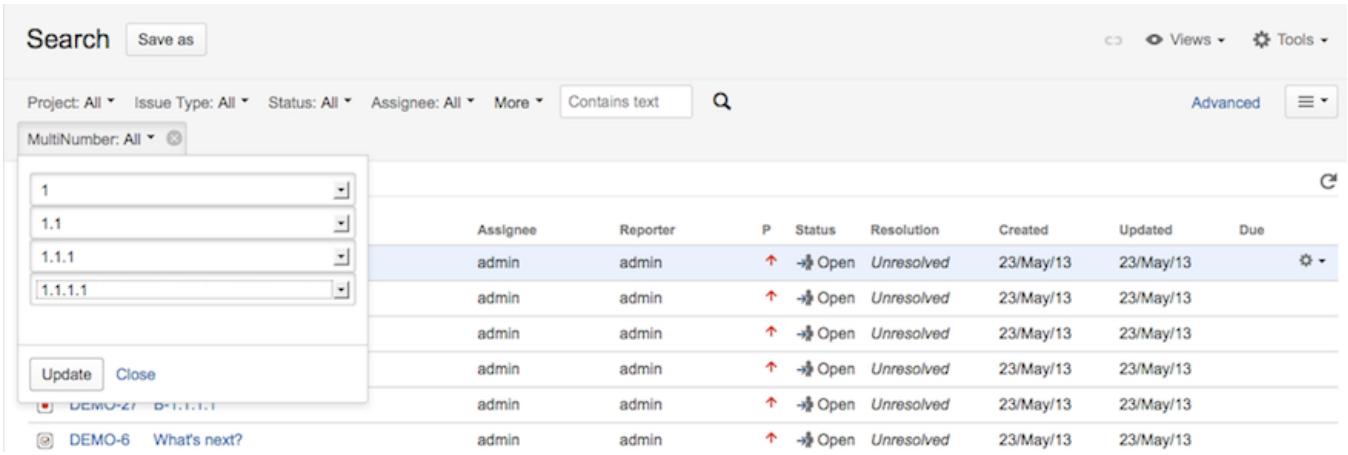


Editing a value is possible through the InLine edit feature or through the normal issue editing interface. To carry out the In line editing, simply point the mouse on the Multi Level Cascade Select value in the issue view screen and click for editing:

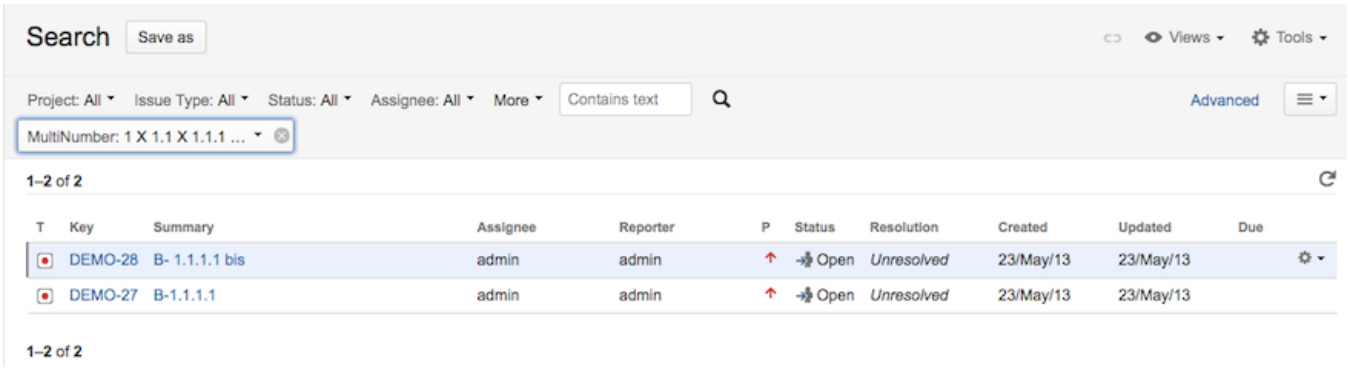


Searcher

To use a Multi Level Cascade value to search issues, in the Issue Navigator add your MLCS custom field to the input filter for the search. A multi select cascade list will be loaded with the list of configured values:



Select the desired value and trigger the search.



1) if you select “Any” for the first level, “Any” will apply to all the levels and the search will return all the issues that have any value for our custom field. Issues with no custom field value will be excluded from the results.
2) if you select “none” for the first level,”none” will apply to all the levels and the search will return all the issues with no value selected for this field. Any issue having any not null value for the custom field will be excluded from the search results.
3) if you select a list of options that ends with a “none” value, the search results will include only the issues with the exact value of the custom field that corresponds to the list of options in input.
Example : I1= 1.1.1, I2=1.1.1.1, I3=1.1.1.2
Search : 1.1.1
Results : I1
4) if you select a list of options that ends with a “Any” value,the search results will include all the issues with the exact value and all the issues that contain children of that value.
Example : I1= 1.1.1, I2=1.1.1.1, I3=1.1.1.2
Search : 1.1.1
Results : I1-I2-I3

Advanced Search

If you directly write your JQL queries, MLCS fields can be searched using the IN operator or a JQL function defined by the add-on, "MultiLevelCascadeOption()".

IN operator

The IN operator produces a result set which is wider than the one we get using the "MultiLevelCascadeOption()" JQL function: it will hit all the cases where the given sequence of option values appears among the selected ones. This is better explained by example:

suppose we have a MLCS custom field "Place" with option structure like

Europe
  > Italy
  > France
America
  > Mexico
  > USA

then the JQL query

```
Place in (America,USA)
```

will retrieve all the issues such that "Place" evaluates to anything that contains "America" followed by "USA", regardless of the depth of the options: this means that, if we add "America" as a child option of "Italy", and "USA" as a child option of the second "America", so that the option structure is made into

Europe
  > Italy
    > America
      > USA
  > France
America
  > Mexico
  > USA

then the issues such that "Place" evaluates to "Europe - Italy - America - USA" will be retrieved (while "MultiLevelCascadeOption(America,USA)" would retrieve only the issues in which "America" is the root option, followed by "USA")

The IN operator alse supports the "None" keyword, so

```
Place in (Europe,None)
```

will retrieve all the issues such that an option of value "Europe" is selected while no deeper option is.

The IN operator can be used also with option IDs, but the None keyword won't work in that case (it will be ignored).

IS operator

The IS operator is only used to find out which issues have the MLCS custom field set to any value, so in our example the two possible queries using IS are:

```
Place is EMPTY
```

which will retrieve all the issues such that "Place" is not set at all;

```
Place is not EMPTY
```

which retrieves all issues such that "Place" is set to any value.

JQL function (DEPRECATED)

The function MultiLevelCascadeOption() gets one or more comma-separated parameters, that can be option IDs and/or values

This function is deprecated in favor of the IN operator and will be removed in future releases

Examples:

- `fieldName IN MultiLevelCascadeOption(100003, 100012)` retrieves the issues where the *fieldName* MLCS custom fields has the option of id "100003" selected on the first level and its child option of id "100012" selected on the second level, regardless of what is selected on the deeper levels
- `fieldName IN MultiLevelCascadeOption(100003, 100012, None)` retrieves the issues where the *fieldName* MLCS custom fields has the option of id "100003" selected on the first level and its child option of id "100012" selected on the second level and no option is selected on the deeper levels
- `fieldName IN MultiLevelCascadeOption(optionValue1, optionValue2)` retrieves the issues where the *fieldName* MLCS custom fields has the option of value "*optionValue1*" selected on the first level and its child option of value "*optionValue2*" selected on the second level, regardless of what is selected on the deeper levels (if we add "none" as last argument, issues where anything is selected on the third level will be excluded)
- `fieldName IN MultiLevelCascadeOption(optionValue1, 100012, None)` retrieves the issues where the *fieldName* MLCS custom fields has the option of value "*optionValue1*" selected on the first level and its child option of id "100012" selected on the second level and no option is selected on the deeper levels

NOTE: even though option values are not JIRA-wise unique, two silbling options cannot get the same value, so the result of `someField in MultilevelCascadeOption("something", "somethingElse", "somethingFurther")` is actually unambiguously defined. There is, though, a rare case of ambiguity: it happens if there exists an MLCS field having two options such that one option's value equals the ID of the other option. In most cases, option hierarchy will establish a unique interpretation for queries, but sometimes a query will actually be ambiguous: in thor really rare cases, a warning dialog will be displayed:

Jql - (Ignore this message if you did not perform Advanced search) - "10003" is both the option id of the option <10003 - Europe> and the option value for the option <10400 - 10003>. Both are root options and subsequent arguments could be sub-opions of both, so there is ambiguity. If you meant "10003" as option id, then the current search is correct (this is always the case if you did a Basic search); otherwise please repeat the query using "10400" instead of "10003"

If you programmaticaly call the function, consider using IDs instead of values if you have to deal with option values which are numbers (IDs are always JIRA-wise unique).

Reports

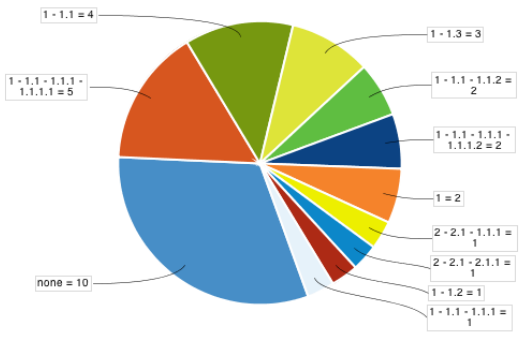
A MLCS field can be used in gadgets and reports, like the filter statistics or the pie chart gadgets:

Two Dimensional Filter Statistics: Filter1

	Assignee
MultiNumber	**admin**
1	2
1 - 1.1	4
1 - 1.1 - 1.1.1	1
1 - 1.1 - 1.1.1 - 1.1.1.1	5
1 - 1.1 - 1.1.1 - 1.1.1.2	2
1 - 1.1 - 1.1.2	2
1 - 1.2	1
1 - 1.3	3
2 - 2.1 - 2.1.1	1
2 - 2.1 - 1.1.1	1
None	10

Showing **11** of **11** statistics. Show less
Filter: Filter1

Pie Chart: Filter1



Total Issues: **32** Statistic Type: **MultiNumber**