REST APIs

Get Options

Since version 4.0.9 the addon defines a REST call which provides information about options

```
GET <jira-base>/rest/multi-level-cascading-select/1/option/getOptions
```

Query parameters:

fieldName (optional)
name of the custom field (retrieve only fields with such name). If this parameter is not specified, options from fields with any name will be retrieved

projectKey (optional)
the key of a project (retrieve only fields associated with such project). If this parameter is not specified, options from fields will be retrieved, regardless of the projects those fields are associeted with

anyType (optional, defaults to "false")
if this is set to true, the method will retrieve not just the options of MultiLevelCascadeSelect fields but of any kind of field which uses options (ie, classes implementing the Option interface). This is known to be well-behaved on Jira Cascade Select fields, but of course there could be no guarantee on all other possible fields which make use of Option implementations

fieldConfigId (optional)
relevant config id. This parameter is optional, but when it is specified, parameters 'fieldName' and 'projectKey'must not be used (parameters are essentially filters for fieldConfigs, so if the ID of a fieldConfig is specified, it makes no sense to use other filters, as ID is unique). If 'anyType=true' is used, then the method will try to get information about options even if the field is not of type MultiLevelCascadeSelect (see 'anyType').

Response:

The options are described by a json structure, containing the fields

optionId - *number*, the (JIRA-wise) unique id number of the option

value - *string*, the value of the option

sequence - *number*, the place that the option occupies among its silblings

disabled - *boolean*, it is true if the option is disabled

fildConfigId - *number*, the ID of the field config to which the option belongs

children - *array of option objects*, the child-options of the option

Example:

```
{
    "optionId": 10504,
    "value": "Europe",
    "sequence": 0,
    "disabled": false,
    "fieldConfigId": 10108,
    "children": [
        {
            "optionId": 10505,
            "value": "Italy",
            "sequence": 0,
            "disabled": false,
            "fieldConfigId": 10108
        },
        {
            "optionId": 10506,
            "value": "France",
            "sequence": 1,
            "disabled": false,
            "fieldConfigId": 10108
        }
    ]
}
```

Add Option

Add a new option as a child of the input parent option. If no parent option is specified, a new root (level 0) option is added.

```
POST <jira-base>/rest/multi-level-cascading-select/1/option/
```

Query parameters:

fieldConfigId (required)
id of the field configuration of the custom field (***warning***: this is not the same thing as the custom field id. If you don't know the field config id, use the getOption REST call without any parameter and you will get the fieldConfigId for all of your MLCS custom fields)

parentOptionId (optional)
if specified, the new option will be added as a child option of the one identified by this parameter. If not specified, the new option will be a root option

optionValue (required)
the value of the new option (this must be different from all sibling options values, not be equal to "Any", not be case-insensitively equal to "none")

Response:

If everything goes well, response status is 200 and response body contains details about the new option, like its ID.

The option is described by a json structure, containing the fields

optionId - *number*, the (JIRA-wise) unique id number of the option

value - *string*, the value of the option

sequence - *number*, the place that the option occupies among its silblings

Example:

```
{
    "optionId": 10505,
    "value": "Italy",
    "sequence": 1
}
```

Delete Option

Remove an option.

```
DELETE <jira-base>/rest/multi-level-cascading-select/1/option/{optionId}
```

where

optionId
id of the option to remove

Query parameters:

fieldConfigId (required)
id of the field configuration of the custom field (***warning***: this is not the same thing as the custom field id. If you don't know the field config id, use the getOption REST call without any parameter and you will get the fieldConfigId for all of your MLCS custom fields)

Response:

If everything goes well, response status is 200 and response body is empty.

Rename Option

Modify the value of an option

```
PUT <jira-base>/rest/multi-level-cascading-select/1/option/{optionId}
```

where

optionId
id of the option to rename

Query parameters:

newValue (required)
the new value of the option (this must be different from all sibling options values, not be equal to "Any", not be case-insensitively equal to "none")

Response:

If everything goes well, response status is 200 and response body contains details about the option.

The option is described by a json structure, containing the fields

optionId - *number*, the (JIRA-wise) unique id number of the option

value - *string*, the value of the option

sequence - *number*, the place that the option occupies among its silblings

Example:

```
{
    "optionId": 10505,
    "value": "Italy",
    "sequence": 1
}
```

Disable Option

Make an option not selectable

```
PUT <jira-base>/rest/multi-level-cascading-select/1/option/{optionId}/disable
```

where

optionId
id of the option to disable

Response:

If everything goes well, response status is 200 and response body contains details about the option.

The option is described by a json structure, containing the fields

optionId - *number*, the (JIRA-wise) unique id number of the option

value - *string*, the value of the option

sequence - *number*, the place that the option occupies among its silblings

Example:

```
{
    "optionId": 10505,
    "value": "Italy",
    "sequence": 1
}
```

Enable Option

Make an option selectable

```
PUT <jira-base>/rest/multi-level-cascading-select/1/option/{optionId}/enable
```

where

optionId
id of the option to enable

Response:

If everything goes well, response status is 200 and response body contains details about the option.

The option is described by a json structure, containing the fields

optionId - *number*, the (JIRA-wise) unique id number of the option

value - *string*, the value of the option

sequence - *number*, the place that the option occupies among its silblings

Example:

```
{
    "optionId": 10505,
    "value": "Italy",
    "sequence": 1
}
```

Move Option

Change the order of the option among its sibling options so that it is repositioned to the given place. The way other options are moved is better explained by an example: suppose we have options in the sequence 0,A - 1,B - 2,C - 3,D - 4,E. If we move B to position 3, we get: 0,A - 1,C - 2,D - 3,B - 4,E.

```
PUT <jira-base>/rest/multi-level-cascading-select/1/option/{optionId}/position
```

where

optionId
id of the option to enable

Query parameters:

fieldConfigId (required)
id of the field configuration of the custom field (***warning***: this is not the same thing as the custom field id. If you don't know the field config id, use the getOption REST call without any parameter and you will get the fieldConfigId for all of your MLCS custom fields)

position (required, must be between 0 and the number of sibling options - 1)
the position where the option must be placed (first position is 0)

Response:

If everything goes well, response status is 200 and response body is empty.

Sort Options

Alphabetically sort the child options of the input parent option. If no parent option is specified, sort the root level options.

```
GET <jira-base>/rest/multi-level-cascading-select/1/option/sort
```

Query parameters:

fieldConfigId (required)
id of the field configuration of the custom field (***warning***: this is not the same thing as the custom field id. If you don't know the field config id, use the getOption REST call without any parameter and you will get the fieldConfigId for all of your MLCS custom fields)

asc (optional)
if this parameter is given, options are sorted in reverse (ascending) order

parentOptionId (optional)
if specified, the child options of the one identified by this parameter will be sorted. If not specified, root options will be sorted

Response:

If everything goes well, response status is 200 and response body is empty.