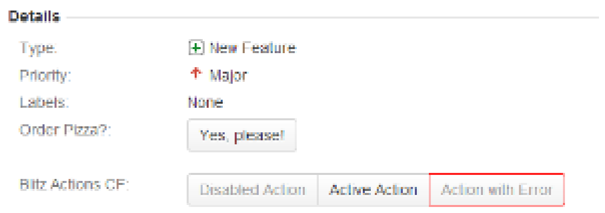
Home

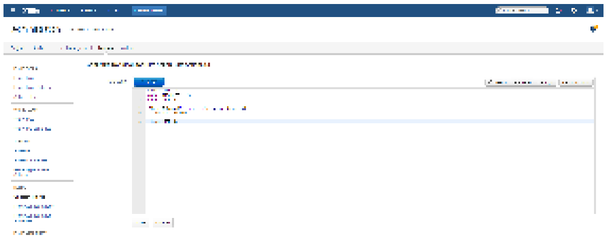
Blitz Actions
BA
Think differently. Work faster with SIL
Extend JIRA actions with your own
Simplify your workflows
Automate interactions with JIRA

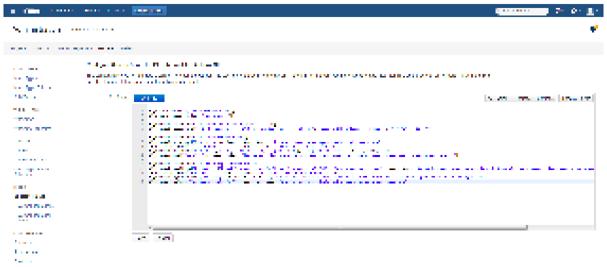
Gallery

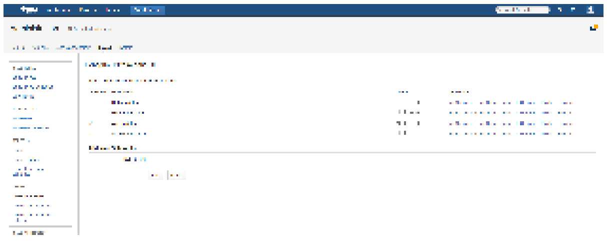

Details

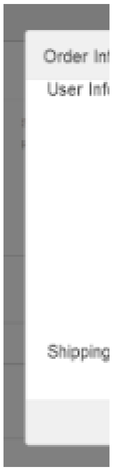
Type: New Feature
Priority: Major
Labels: None
Order Pizza?: Yes, please!

Blitz Actions CF: Disabled Action | Active Action | Action with Error









The Blitz Actions plugin introduces custom made actions to be performed in the view screen of the issue.

The plugin extends the actions available for JIRA issues with custom actions created using Simple Issue Language (SIL) - the same language used in JJUPIN and KCF plugins.

A blitz action has same activation structure as a (workflow) transition and it can be performed regardless of the status of the issue, provided that the condition of the action is fulfilled.

In short the plugin defines:

- A custom field presented as a set of buttons
- Each button is associated with an action
- An action has 3 elements
  - a condition script, that is the rule that enables, disables or hides the button
  - an optional screen
  - an action script, that describes both the validation of the screen and the actions to be performed
- All action elements (condition, action script and screen) are described using SIL.

Blitz Actions plugin requires minutes to create a custom action. Do you know a faster way?

Recently Updated

KBA30
Feb 28, 2017 • attached by Alexandru Geageac

Kepler Blitz Actions 3.0
Feb 24, 2017 • created by Confluence Administrator

KBA30
Feb 24, 2017 • attached by Confluence Administrator

What should I do if I installed an incompatible version?
Dec 02, 2015 • created by Alexandra Topoloaga

Install notes for JIRA 7
Nov 16, 2015 • created by Alexandra Topoloaga

Requirements
Nov 16, 2015 • created by Alexandra Topoloaga

The Action Script
Apr 15, 2015 • created by Silviu Burcea

Configuring a Blitz Actions Custom Field
Mar 05, 2015 • created by Alexandra Topoloaga

image2015-3-5 10:34:39.png
Mar 05, 2015 • attached by Alexandra Topoloaga

Backup and restore
Feb 19, 2015 • created by Alexandru Geageac

The Administration Page
Sep 18, 2014 • created by Maria Cirtog

Licensing
Sep 18, 2014 • created by Maria Cirtog

Installation via Atlassian Universal Plugin Manager
Sep 18, 2014 • created by Maria Cirtog

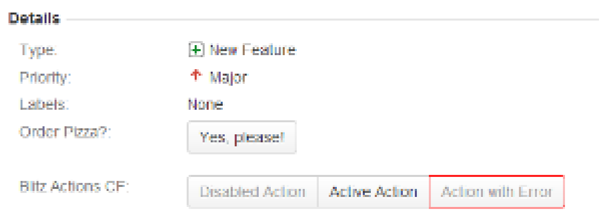
Manual Uninstall
Sep 15, 2014 • created by Adrian Iasinschi

Uninstall via Atlassian Universal Plugin Manager
Sep 15, 2014 • created by Adrian Iasinschi

Kepler Blitz Actions Documentation

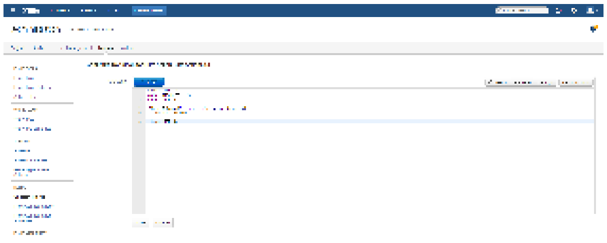
BA
Blitz Actions
Think differently. Work faster with SIL
› Extend JIRA actions with your own
› Simplify your workflows
› Automate interactions with JIRA

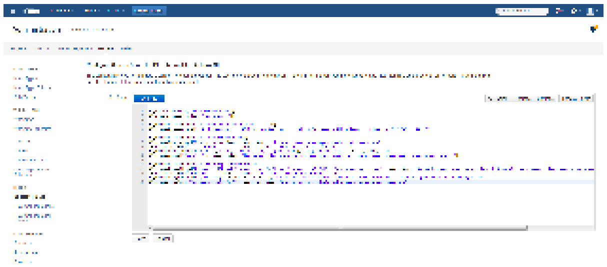
Gallery

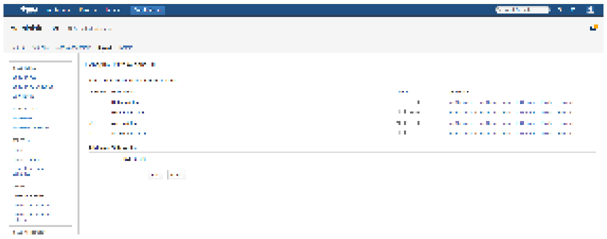


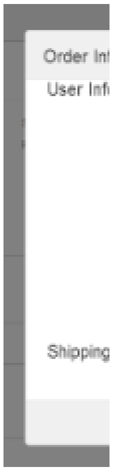
Details

Type:	New Feature
Priority:	Major
Labels:	None
Order Pizza?:	Yes, please!
Blitz Actions CF:	Disabled Action · Active Action · Action with Error









The Blitz Actions plugin introduces custom made actions to be performed in the view screen of the issue.

The plugin extends the actions available for JIRA issues with custom actions created using Simple Issue Language (SIL) - the same language used in JJUPIN and KCF plugins.

A blitz action has same activation structure as a (workflow) transition and it can be performed regardless of the status of the issue, provided that the condition of the action is fulfilled.

In short the plugin defines:

- A custom field presented as a set of buttons
- Each button is associated with an action
- An action has 3 elements
    - a condition script, that is the rule that enables, disables or hides the button
    - an optional screen
    - an action script, that describes both the validation of the screen and the actions to be performed
- All action elements (condition, action script and screen) are described using SIL.

Blitz Actions plugin requires minutes to create a custom action. Do you know a faster way?

Recently Updated

KBA30
Feb 28, 2017 • attached by Alexandru Geageac

Kepler Blitz Actions 3.0
Feb 24, 2017 • created by Confluence Administrator

KBA30
Feb 24, 2017 • attached by Confluence Administrator

What should I do if I installed an incompatible version?
Dec 02, 2015 • created by Alexandra Topoloaga

Install notes for JIRA 7

Nov 16, 2015 • created by Alexandra Topoloaga

Requirements
Nov 16, 2015 • created by Alexandra Topoloaga

The Action Script
Apr 15, 2015 • created by Silviu Burcea

Configuring a Blitz Actions Custom Field
Mar 05, 2015 • created by Alexandra Topoloaga

image2015-3-5 10:34:39.png
Mar 05, 2015 • attached by Alexandra Topoloaga

Backup and restore
Feb 19, 2015 • created by Alexandru Geageac

The Administration Page
Sep 18, 2014 • created by Maria Cirtog

Licensing
Sep 18, 2014 • created by Maria Cirtog

Installation via Atlassian Universal Plugin Manager
Sep 18, 2014 • created by Maria Cirtog

Manual Uninstall
Sep 15, 2014 • created by Adrian Iasinschi

Uninstall via Atlassian Universal Plugin Manager
Sep 15, 2014 • created by Adrian Iasinschi

Additional Documentation

Tutorials

You can find out more examples in our extra-documentation space: Tutorials & Recipes.

User Guide

It often happens that our customers require custom transitions that will change some values on an issue, or take action based on those values without advancing it through the workflow. This usually required identical transitions to be created for every step of the workflow, which is time-consuming as well as difficult to maintain.

We decided that we could use our own Simple Issue Language 4.0 to create custom actions, similar to transitions, that are flexible and also reusable, making maintenance and development a breeze.

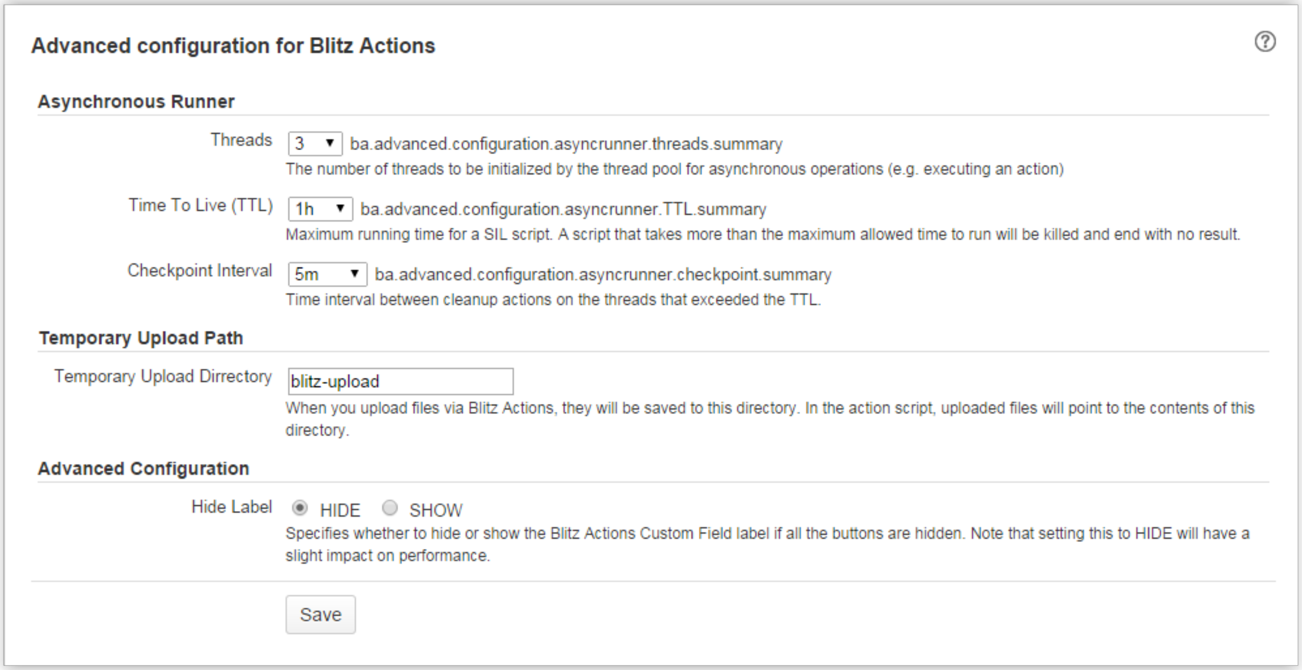
The actions are based on a custom field. This makes them easy to use and also benefit from the already existing Custom Field Context configuration available with JIRA.

The Blitz Actions use 3 SIL scripts each:

1. **The condition script**. This decides whether the action is available for the user to execute, or if it is visible to the user but not executable, or if it is simply hidden with no visual indication that it exists.
2. **The screen script**. This decides if an action requires some user input before the action is executed. It allows the administrator to generate a form with fields that will provide input to the actual execution of the action.
3. **The action script.** This is the script that does all the work after all input has been gathered.

The Administration Page

Starting with version 2.5, Blitz Actions also provides a configuration page which can be found in Administration-> Add-ons-> Blitz Actions.


Advanced configuration for Blitz Actions

Asynchronous Runner

Threads: 3 ▾ ba.advanced.configuration.asyncrunner.threads.summary
The number of threads to be initialized by the thread pool for asynchronous operations (e.g. executing an action)

Time To Live (TTL): 1h ▾ ba.advanced.configuration.asyncrunner.TTL.summary
Maximum running time for a SIL script. A script that takes more than the maximum allowed time to run will be killed and end with no result.

Checkpoint Interval: 5m ▾ ba.advanced.configuration.asyncrunner.checkpoint.summary
Time interval between cleanup actions on the threads that exceeded the TTL.

Temporary Upload Path

Temporary Upload Directory: blitz-upload
When you upload files via Blitz Actions, they will be saved to this directory. In the action script, uploaded files will point to the contents of this directory.

Advanced Configuration

Hide Label: ● HIDE ○ SHOW
Specifies whether to hide or show the Blitz Actions Custom Field label if all the buttons are hidden. Note that setting this to HIDE will have a slight impact on performance.

Here you can configure the following parameters:

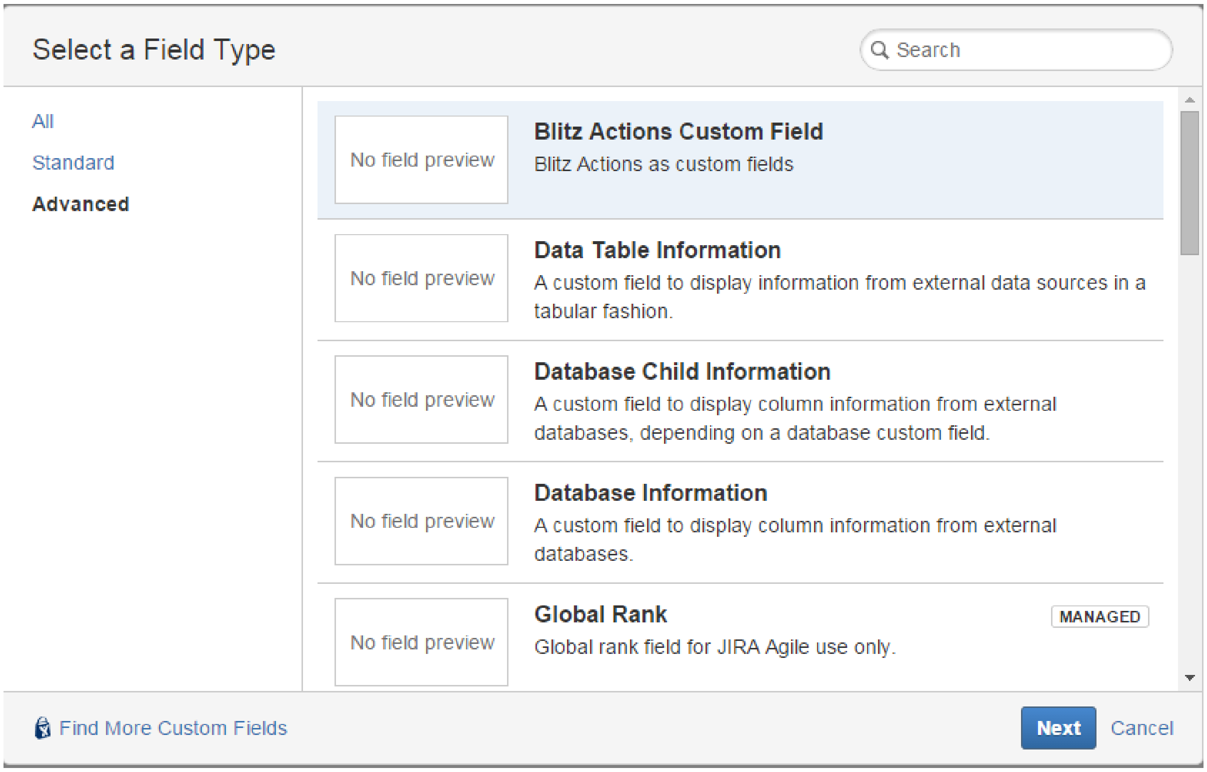
- **Asynchronous Runne**r - When an action is called, the script execution is passed to a thread pool. These parameters define the thread pool behavior.
  - Threads - The number of threads in the pool.
  - TTL - Maximum allowed time for a script to run. If the execution takes any longer than the TTL, it will be halted.
  - Checkpoint Interval - When a script is halted, there may be leftover data. This parameter configures the frequency for the cleanup actions on this data.
- **Temporary Upload Path**
  - Temporary Upload Directory - When using file upload inputs on a Blitz Screen, the files are uploaded asynchronously, immediately after they were chosen. You cannot execute the action until all files were uploaded. Naturally, the location where the files are uploaded is configurable via this parameter. If given an absolute path, it will be used as-is; otherwise, the path will be considered relative to the kepler folder in your JIRA Home directory.
- **Advanced Configuration**
  - Hide Label (for versions 2.0.8+ and 2.6.1+) - Specifies the behavior of a Blitz Actions custom field when all the buttons are hidden. If set to SHOW, the label will be shown along with the message "No actions available". If set to HIDE, will not display neither the label nor the message. Note that setting this to HIDE may slightly impact performance.

Warning
There is no cleanup action performed on the Temporary Upload Directory. It is the user's responsibility to cleanup after the script is done. See deleteFile. Note that users can upload files and then cancel the execution of the action. This will leave the uploaded files on the disk and may require regular cleanup.

Configuring a Blitz Actions Custom Field

Adding the Blitz Actions Custom Field

Adding a new Blitz Actions Custom Field is as easy as adding any other custom field. Just go to **Administration -> Custom Fields -> Add Custom Field** and select the **Blitz Actions Custom Field** from the list and follow the on-screen instructions.



No field preview

Blitz Actions Custom Field

Blitz Actions as custom fields

No field preview

Data Table Information

A custom field to display information from external data sources in a tabular fashion.

No field preview

Database Child Information

A custom field to display column information from external databases, depending on a database custom field.

No field preview

Database Information

A custom field to display column information from external databases.

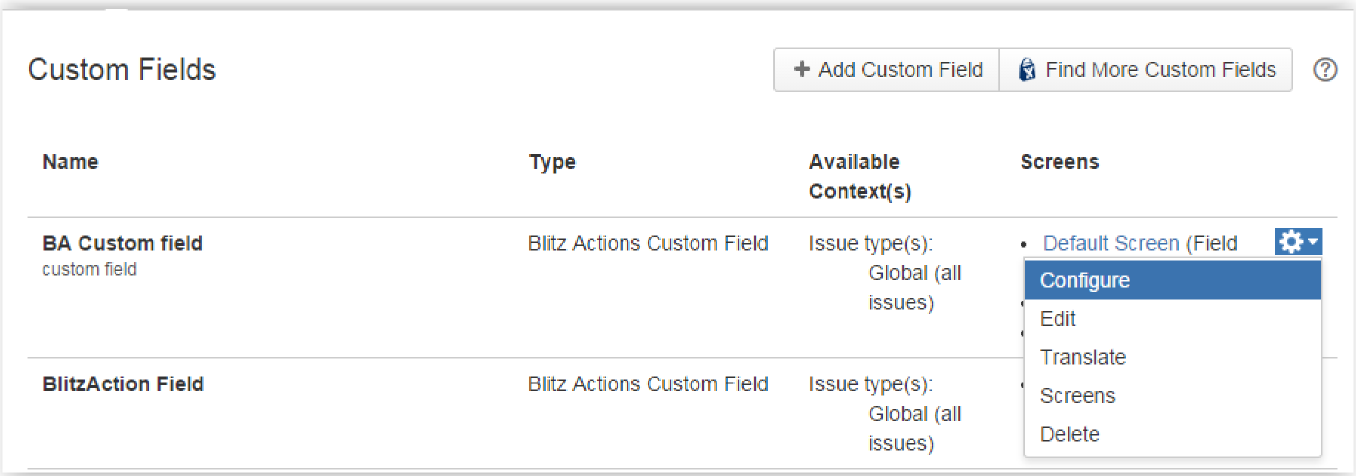
No field preview

Global Rank MANAGED

Global rank field for JIRA Agile use only.

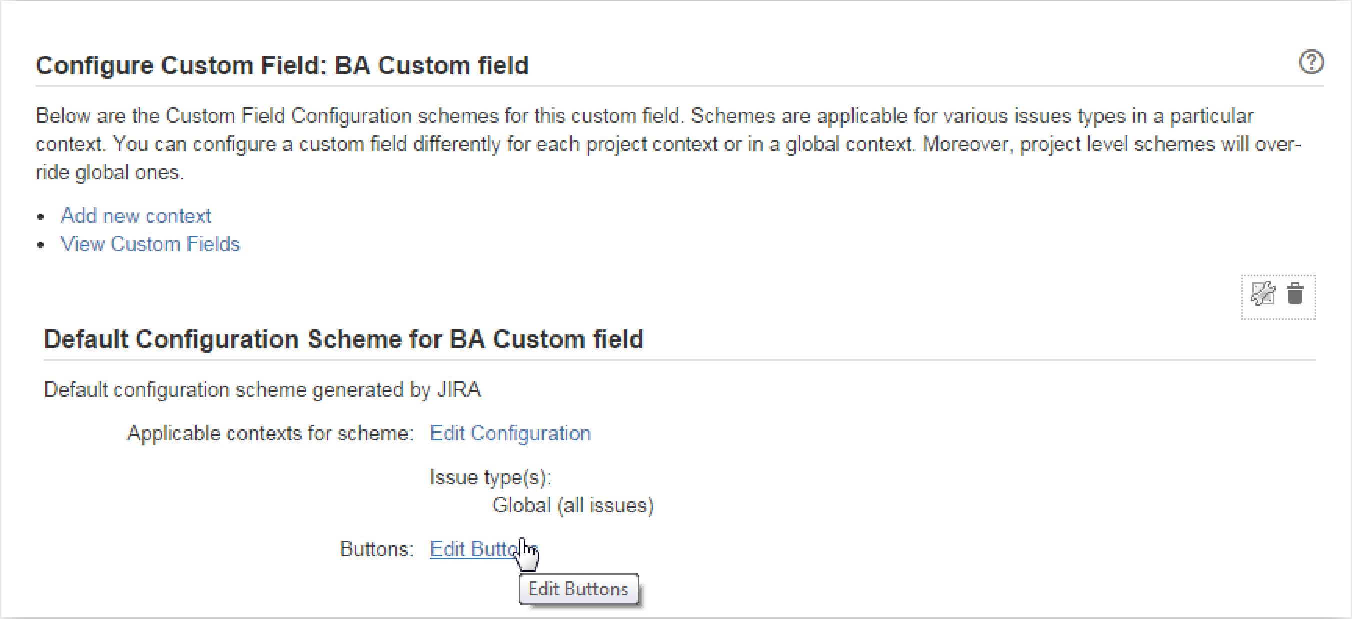
Adding Actions

After you have added the custom field, it's time to add some actions. From the **Custom Fields** screen, select **Configure** for the Blitz Action custom field.



Name	Type	Available Context(s)	Screens
BA Custom field custom field	Blitz Actions Custom Field	Issue type(s): Global (all issues)	• Default Screen (Field
BlitzAction Field	Blitz Actions Custom Field	Issue type(s): Global (all issues)	•

To manage the actions for a given context, click the **Edit Buttons** link from the configuration screen.



Configure Custom Field: BA Custom field

Below are the Custom Field Configuration schemes for this custom field. Schemes are applicable for various issues types in a particular context. You can configure a custom field differently for each project context or in a global context. Moreover, project level schemes will override global ones.

- Add new context
- View Custom Fields

Default Configuration Scheme for BA Custom field

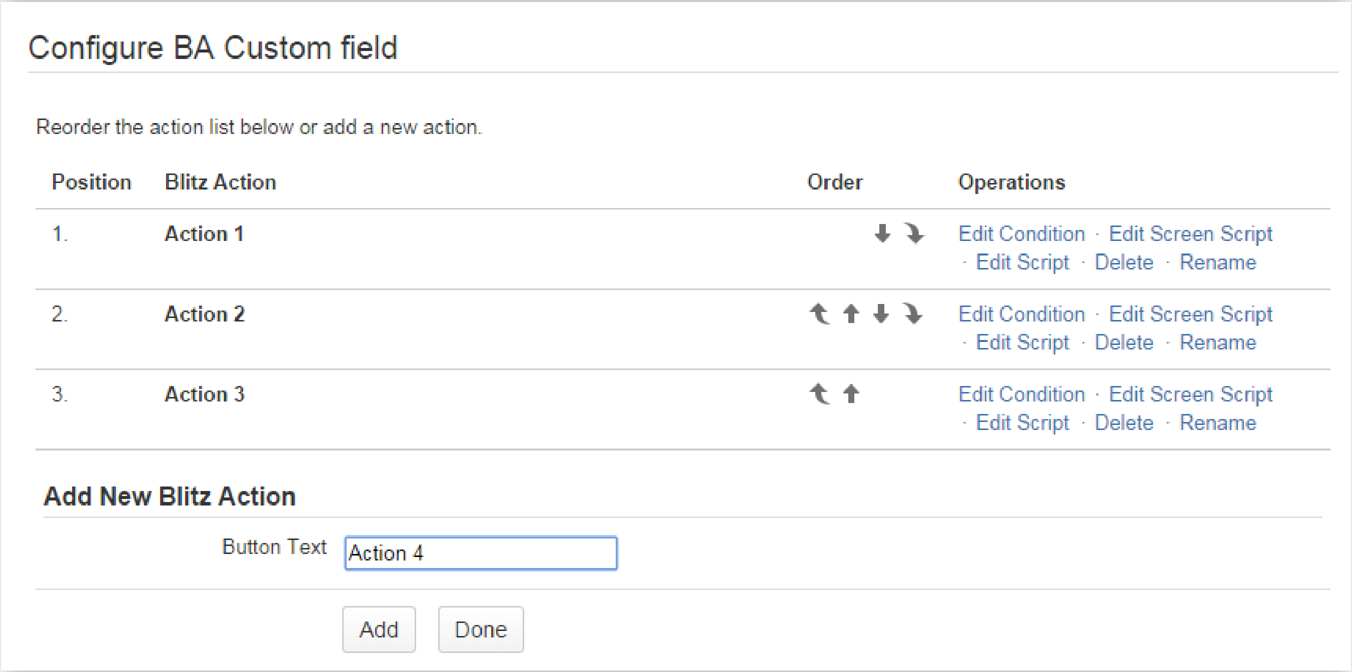
Default configuration scheme generated by JIRA

Applicable contexts for scheme: Edit Configuration

Issue type(s):
Global (all issues)

Buttons: Edit Buttons

You will be presented with a screen similar to other option-based custom fields.



Configure BA Custom field

Reorder the action list below or add a new action.

Position	Blitz Action	Order	Operations
1.	**Action 1**	⬇ ↘	Edit Condition · Edit Screen Script · Edit Script · Delete · Rename
2.	**Action 2**	↖ ⬆ ⬇ ↘	Edit Condition · Edit Screen Script · Edit Script · Delete · Rename
3.	**Action 3**	↖ ⬆	Edit Condition · Edit Screen Script · Edit Script · Delete · Rename

Add New Blitz Action

Button Text: Action 4

Here you can add new actions, delete existing actions and reorder them to suit your needs.

Once you've added an action, it's time to edit the scripts behind it by clicking one of the links available for each one.

See these pages for detailed instructions on each script.

- The Condition Script
- The Screen Script
- The Action Script
- _routine_template

The Condition Script

The condition script allows you to decide whether an action will be available for the user, or if it will be visible but disabled, or if it will simply be hidden from the user.

By default, when adding a new action, it will have the following default condition script.

```
number ENABLED = 1;
number DISABLED = 2;
number HIDDEN = 3;

return ENABLED;
```

The script should return one of the predefined variables: ENABLED, DISABLE or HIDDEN.

- Enabled actions are available for the user to execute.
- Disabled means that the button for the action will be visible, but it will be disabled, preventing the user from executing the action.
- Hidden actions will not be visible at all and thus not available for the user to execute.

Of course, since we are using SIL, the return value of the script can be conditional.

```
number ENABLED = 1;
number DISABLED = 2;
number HIDDEN = 3;

if(<some_condition>){
 return ENABLED;
}
if(<some_other_condition>){
 return DISABLED;
}
return HIDDEN;
```

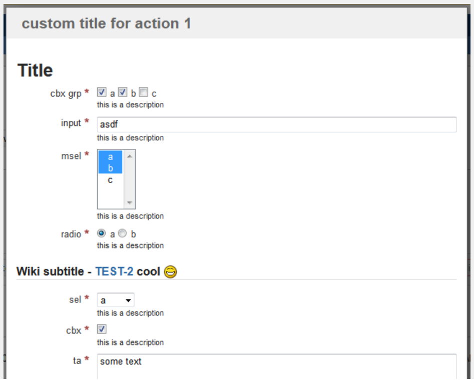
The Screen Script

On this page:

The screen script allows you to determine if an action requires additional input and to build a form where the user can fill in the necessary data.

Input types

Blitz Actions provides a variety of input fields to suit all your needs.

Category	Input Type	
Content	HTML content	
	Wiki content	
No options, single value	text	
	textarea	
	single checkbox	
	date picker	

custom title for action 1

Title

cbx grp * ☑ a ☑ b □ c
this is a description

input * asdf
this is a description

msel * a b c
this is a description

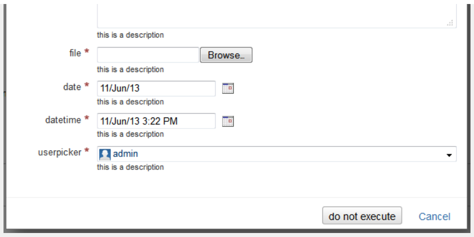
radio * ● a ○ b
this is a description

Wiki subtitle - TEST-2 cool 😄

sel * a
this is a description

cbx * ☑
this is a description

ta * some text

	date time picker	 this is a description file * this is a description date * 11/Jun/13 this is a description datetime * 11/Jun/13 3:22 PM this is a description userpicker * admin this is a description
	user picker	
No options, multiple values	file upload	
Multiple options, one selectable value	select list	
	radio group	
Multiple options, multiple selectable values	multi select list	
	checkbox group	

Creating the screen (v2.0.8+ for JIRA 5 and 2.6.1+ for JIRA 6)

The old method of creating screens was a bit clumsy and unintuitive, so starting from version 2.0.8 and 2.6.1, all you have to do to add an input on the screen is call the respective routine for that input type. No more adding to a huge array and returning it. Note that returning the array will revert to using the old method which does not support newer input types.

See the specific page for each routine that can create an input:

- BA_createCheckboxGroup
- BA_createDatePicker
- BA_createDateTimePicker
- BA_createFileUpload
- BA_createHtmlContent
- BA_createInput
- BA_createMultiSelectList
- BA_createRadioGroup
- BA_createSelectList
- BA_createSingleCheckbox
- BA_createTextArea
- BA_createUserPicker
- BA_createWikiContent

Additional Screen Building Routines

> **Availability**
> Available since v2.0.8 for JIRA 5 and v2.6.1 for JIRA 6).

In addition to adding controls to the screen, Blitz Actions provides some more routines to control the way your screen is displayed (eg. changing the dialog title, the text on the submit button).

See the specific page for each routine that can interact with the screen:

- BA_setActionTitle
- BA_setExecuteButtonText

Creating the screen (up to v2.0.7 for JIRA 5 and 2.6.0 for JIRA 6)

> **Deprecation**
> This method of creating the screen is deprecated starting with version 2.0.8 (for JIRA 5) and 2.6.1 (for JIRA 6). New input types added in versions 2.0.8+ and 2.6.1+ will not work using this method. Scroll down to the new method of creating screens.

The script must return an array of strings, containing parameters for each field that should be displayed. The array will be build from concatenated sub-lists, each of which will describe a field to be shown on screen. If, however, the script does not return a value, or the value is empty, no screen will be shown.

To help create the string array that needs to be returned, we have created some plugin-specific SIL routines with user-friendly syntax that will return the sub-list for each field.

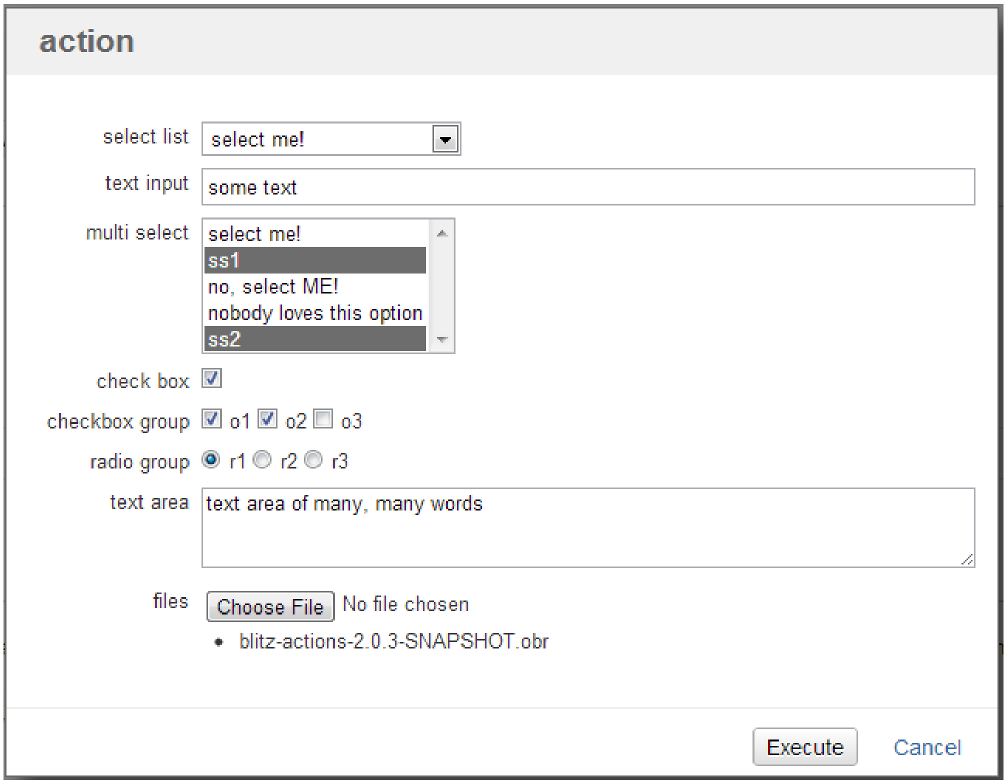
Category	Input Type	SIL routine	Parameter types
No options, single value	TEXT	BA_createInput(label, defaultValue, isDisabled)	label : string defaultValue: string isDisabled : boolean
	TEXT_AREA	BA_createTextArea(label, defaultvalue, isDisabled [, rows])	label : string defaultValue: string isDisabled : boolean rows: number. Optional parameter to specify how tall the text area should be (in lines). If not specified, default will be 3.
	SINGLE_CHECKBOX	BA_createSingleCheckbox(label, isSelected, isDisabled)	label : string isSelected: boolean isDisabled : boolean
No options, multiple values	FILE_UPLOAD (since v. 2.5)	BA_createFileUpload(label, isDisabled)	label: string isDisabled : boolean
Multiple options, one selectable value	SELECT_LIST	BA_createSelectList(label, options, defaultValue, isDisabled)	label : string options: string [] defaultValue: string (must match one of the options or empty string) isDisabled : boolean
	RADIO_GROUP	BA_createRadioGroup(label, options, defaultValue, isDisabled)	
Multiple options, multiple selectable values	MULTI_SELECT_LIST	BA_createMultiSelectList(label, options, defaults, isDisabled)	label : string options: string [] defaults: string [] (must match some of the options or empty array) isDisabled : boolean
	CHECKBOX_GROUP	BA_createCheckboxGroup(label, options, defaults, isDisabled)	

Note
Don't forget to gather all the sub-lists into a single array and return it at the end of the script.

Examples (up to v2.0.7 for JIRA 5 and 2.6.0 for JIRA 6)

```
boolean isDisabled = false;
string [] ret = BA_createSelectList("select list", {"", "select me!",
"ss1", "no, select ME!", "nobody loves this option", "ss2"}, "",
isDisabled);
ret = arraysConcat(ret, BA_createInput("text input", "some text",
isDisabled));
ret = arraysConcat(ret, BA_createMultiSelectList("multi select", {"select
me!", "ss1", "no, select ME!", "nobody loves this option", "ss2"}, {"ss1",
"ss2"}, isDisabled));
ret = arraysConcat(ret, BA_createSingleCheckbox("check box", true,
isDisabled));
ret = arraysConcat(ret, BA_createCheckboxGroup("checkbox group", {"o1",
"o2", "o3"}, {"o1", "o2"}, isDisabled));
ret = arraysConcat(ret, BA_createRadioGroup("radio group", {"r1", "r2",
"r3"}, "r1", isDisabled));
ret = arraysConcat(ret, BA_createTextArea("text area", "text area of many,
many words", isDisabled));
ret = arraysConcat(ret, BA_createFileUpload("files", isDisabled));
return ret;
```

The above script will create one input of each type on the screen.



action

select list: select me!

text input: some text

multi select: select me! / ss1 / no, select ME! / nobody loves this option / ss2

check box ☑

checkbox group ☑ o1 ☑ o2 □ o3

radio group ● r1 ○ r2 ○ r3

text area: text area of many, many words

files: No file chosen

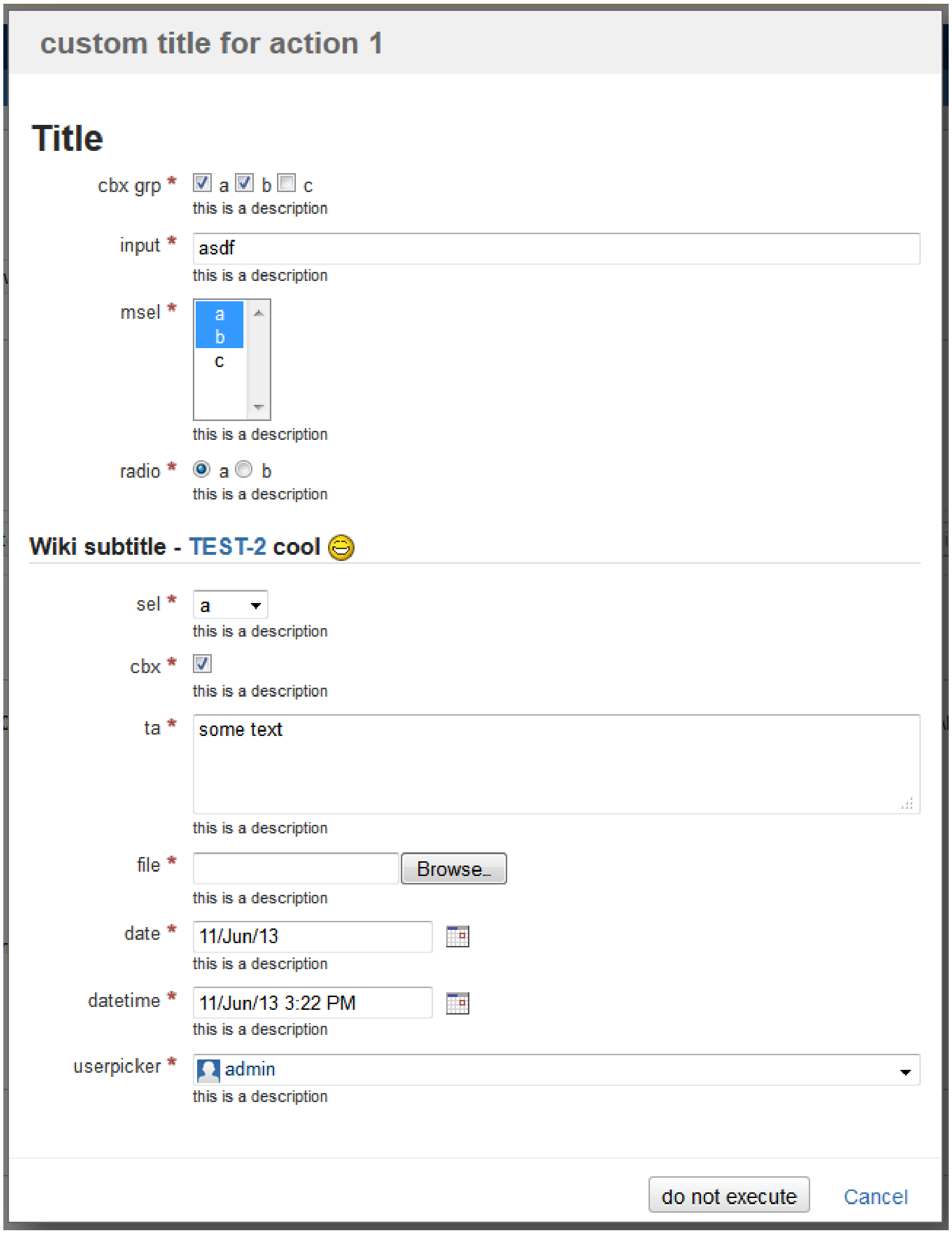
- blitz-actions-2.0.3-SNAPSHOT.obr

Examples (v2.0.8+ for JIRA 5 and 2.6.1+ for JIRA 6)

Example screen script

```
boolean isDisabled = false;
boolean isRequired = true;
string badesc = "this is a description";
BA_createHtmlContent("<h1>Title</h1>");
BA_createCheckboxGroup("cbx grp", {"a", "b", "c"}, {"a", "b"}, isDisabled, isRequired, badesc);
BA_createInput("input", "asdf", isDisabled, isRequired, badesc);
BA_createMultiSelectList("msel", {"a", "b", "c"}, {"a", "b"}, isDisabled, isRequired, badesc);
BA_createRadioGroup("radio", {"a", "b"}, "a", isDisabled, isRequired, badesc);
BA_createWikiContent("h2. Wiki subtitle - TEST-2 cool :D");
BA_createSelectList("sel", {"a", "b"}, "a", isDisabled, isRequired, badesc);
BA_createSingleCheckbox("cbx", true, isDisabled, isRequired, badesc);
BA_createTextArea("ta", "some text", isDisabled, 3, isRequired, badesc);
BA_createFileUpload("file", isDisabled, isRequired, badesc);
date d = currentDate();
BA_createDatePicker("date", d, isDisabled, isRequired, badesc);
BA_createDateTimePicker("datetime", d, isDisabled, isRequired, badesc);
BA_createUserPicker("userpicker", "admin", isDisabled, isRequired, badesc);

BA_setActionTitle("custom title for action 1");
BA_setExecuteButtonText("do not execute");
```



custom title for action 1

Title

cbx grp * ☑ a ☑ b □ c
this is a description

input * asdf
this is a description

msel * a b c
this is a description

radio * ● a ○ b
this is a description

Wiki subtitle - TEST-2 cool 😁

sel * a
this is a description

cbx * ☑
this is a description

ta * some text
this is a description

file *
this is a description

date * 11/Jun/13
this is a description

datetime * 11/Jun/13 3:22 PM
this is a description

userpicker * admin
this is a description

Changes in version 2.0.8 and 2.6.1

There were some major improvements in versions 2.0.8 (for JIRA 5) and 2.6.1 (for JIRA 6).

We realized that the old way of building screens by appending elements to an array using a specific order was a bit unintuitive, so we decided to remove that altogether. All you have to do now to add a field to the screen is call one of the Input Type Routines.

This posed some compatibility issues due to the fact that we added a lot of new field types, but did not want to encourage the use of the old behavior by providing backward compatibility for the new types. Therefore, if you'd like to use any of the new features released with version 2.0.8 and 2.6.1, you'll need to modify the screen scripts to stop returning a value.

before

```
boolean isDisabled = false;
string [] ret = BA_createSelectList("select list", {"", "select me!",
"ss1", "no, select ME!", "nobody loves this option", "ss2"}, "",
isDisabled);
ret = arraysConcat(ret, BA_createInput("text input", "some text",
isDisabled));
ret = arraysConcat(ret, BA_createMultiSelectList("multi select", {"select
me!", "ss1", "no, select ME!", "nobody loves this option", "ss2"}, {"ss1",
"ss2"}, isDisabled));
ret = arraysConcat(ret, BA_createSingleCheckbox("check box", true,
isDisabled));
return ret;
```

after

```
boolean isDisabled = false;
BA_createSelectList("select list", {"", "select me!", "ss1", "no, select
ME!", "nobody loves this option", "ss2"}, "", isDisabled);
BA_createInput("text input", "some text", isDisabled);
BA_createMultiSelectList("multi select", {"select me!", "ss1", "no, select
ME!", "nobody loves this option", "ss2"}, {"ss1", "ss2"}, isDisabled);
BA_createSingleCheckbox("check box", true, isDisabled);
```

So the new changes **are backwards compatible**, meaning that your old scripts will still work the way they used to. You **do not need to modify** t he screen scripts **unless you plan on using any of the new features.**

New features in versions 2.0.8 and 2.6.1

More field types:

- **date picker** using BA_createDatePicker
- **date time picker** using BA_createDateTimePicker
- **user picker** using BA_createUserPicker

Ability to add **content** to the screen to provide more information or visual aid.

- BA_createHtmlContent will append HTML content to the screen
- BA_createWikiContent will render wiki-formatted content and append it to the screen

More control over the screen with Dialog Control Routines:

- BA_setActionTitle will change the title of the dialog
- BA_setExecuteButtonText will change the text displayed on the submit button of the dialog.

Ability to mark fields as **required** with a dark-red asterisk, specified by an additional parameter to the routine creating the field.

Ability to add **descriptions** to the field, specified by an additional parameter to the routine creating the field.

New **BA_getDateValue** to be used along with the date picker and date time picker in the action script.

Dialog Control Routines

These are the routines that will interact with the appearance of the dialog.

- BA_setActionTitle

- BA_setExecuteButtonText

BA_setActionTitle

Availability
This routine is available since **Blitz-Actions 2.0.8 and 2.6.1 .**

Syntax:

BA_setActionTitle(value)

Description:

Overrides the default dialog title.

The last call of this routine will override any previous calls.

Parameters:

Parameter name	Type	Required	Description
value	string	Yes	The new action title

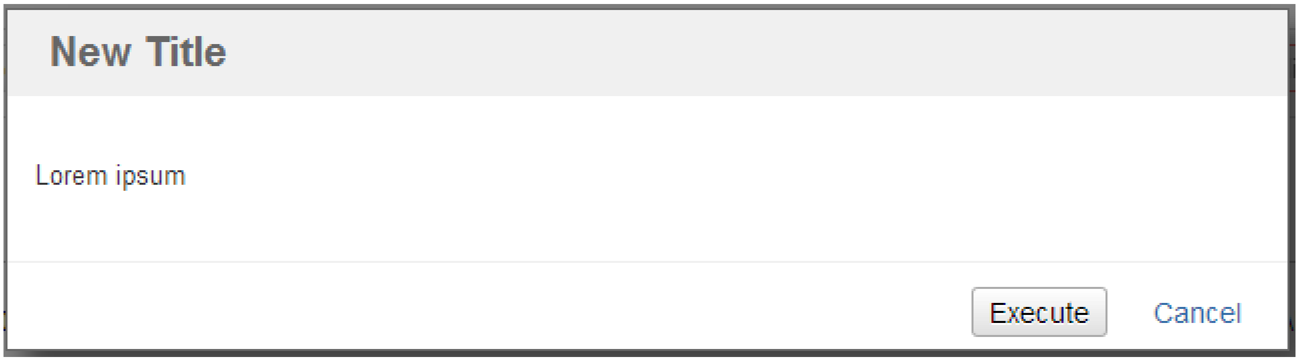
Return type:

The returned value has no meaning.

Examples:

Example 1:

```
BA_createHtmlContent("Lorem ipsum");
BA_setActionTitle("New Title");
```



New Title

Lorem ipsum

BA_setExecuteButtonText

Availability
This routine is available since **Blitz-Actions 2.0.8 and 2.6.1 .**

Syntax:

BA_setExecuteButtonText(value)

Description:

Overrides the default text on the dialog submit button.

The last call of this routine will override any previous calls.

Parameters:

Parameter name	Type	Required	Description
value	string	Yes	The new text to show on the button

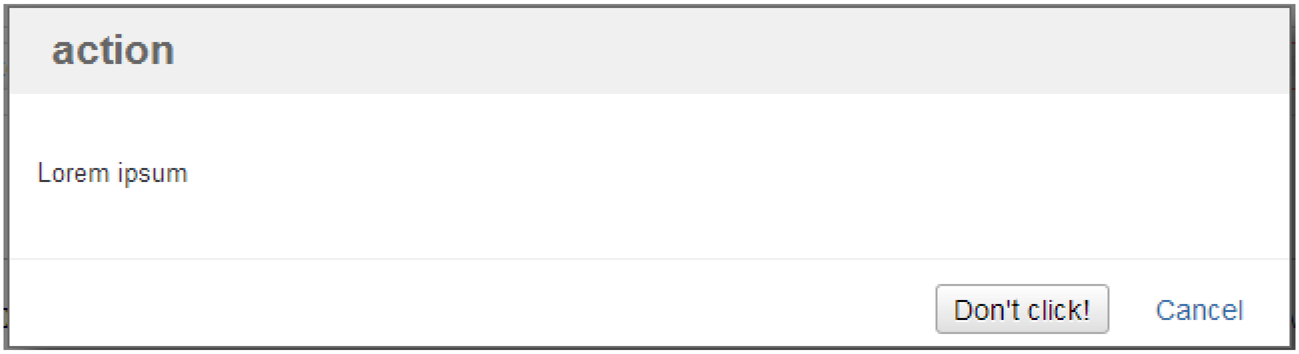
Return type:

The returned value has no meaning.

Examples:

Example 1:

```
BA_createHtmlContent("Lorem ipsum");
BA_setExecuteButtonText("Don't click!");
```



Input Type Routines

These are the routines that will create an input control on the screen when called.

- BA_createCheckboxGroup
- BA_createDatePicker
- BA_createDateTimePicker
- BA_createFileUpload
- BA_createHtmlContent
- BA_createInput
- BA_createMultiSelectList
- BA_createRadioGroup
- BA_createSelectList
- BA_createSingleCheckbox
- BA_createTextArea
- BA_createUserPicker
- BA_createWikiContent

BA_createCheckboxGroup

Syntax:

BA_createCheckboxGroup(label, options, defaultValues, isDisabled)

Since version 2.0.8 and 2.6.1:

BA_createCheckboxGroup(label, options, defaultValues, isDisabled[, isRequired, fieldDescription])

Description:

Creates a checkbox group.

Parameters:

Parameter name	Type	Required	Description
label	string	Yes	The label of the field
options	string []	Yes	The list of selectable options.
defaultValue	string []	Yes	A sub-list of the options provided or an empty array
isDisabled	boolean	Yes	Specifies whether the field is read-only or not
isRequired	boolean	No	Specifies if the field is should be marked as required with a dark red asterisk. Note that marking the field as required does not add any validation.
fieldDescription	string	No	A description of the field to be displayed immediately under the input box.

Return type:

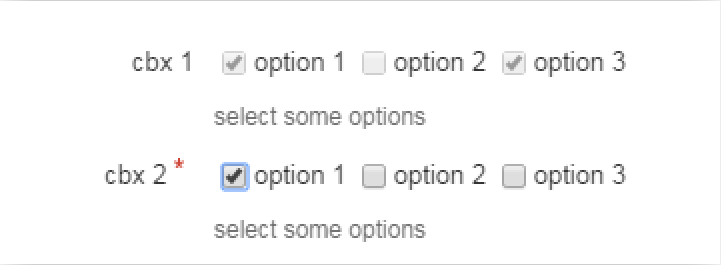
string []

Examples:

Example 1:

```
BA_createCheckboxGroup("cbx 1", {"option 1", "option 2", "option 3"},
{"option 1", "option 3"}, true, false, "select some options");
BA_createCheckboxGroup("cbx 2", {"option 1", "option 2", "option 3"},
"option 1", false, true, "select some options");
```

—



cbx 1 ☑ option 1 □ option 2 ☑ option 3

select some options

cbx 2 * ☑ option 1 □ option 2 □ option 3

select some options

BA_createDatePicker

Availability
This routine is available since **Blitz-Actions 2.0.8 and 2.6.1 .**

Syntax:

BA_createDatePicker(label, defaultValue, isDisabled, isRequired, fieldDescription)

Description:

Creates a date picker.

Parameters:

Parameter name	Type	Required	Description
label	string	Yes	The label of the field

defaultValue	date	Yes	A default value or an empty string
isDisabled	boolean	Yes	Specifies whether the field is read-only or not
isRequired	boolean	Yes	Specifies if the field is should be marked as required with a dark red asterisk. Note that marking the field as required does not add any validation.
fieldDescription	string	Yes	A description of the field to be displayed immediately under the input box.

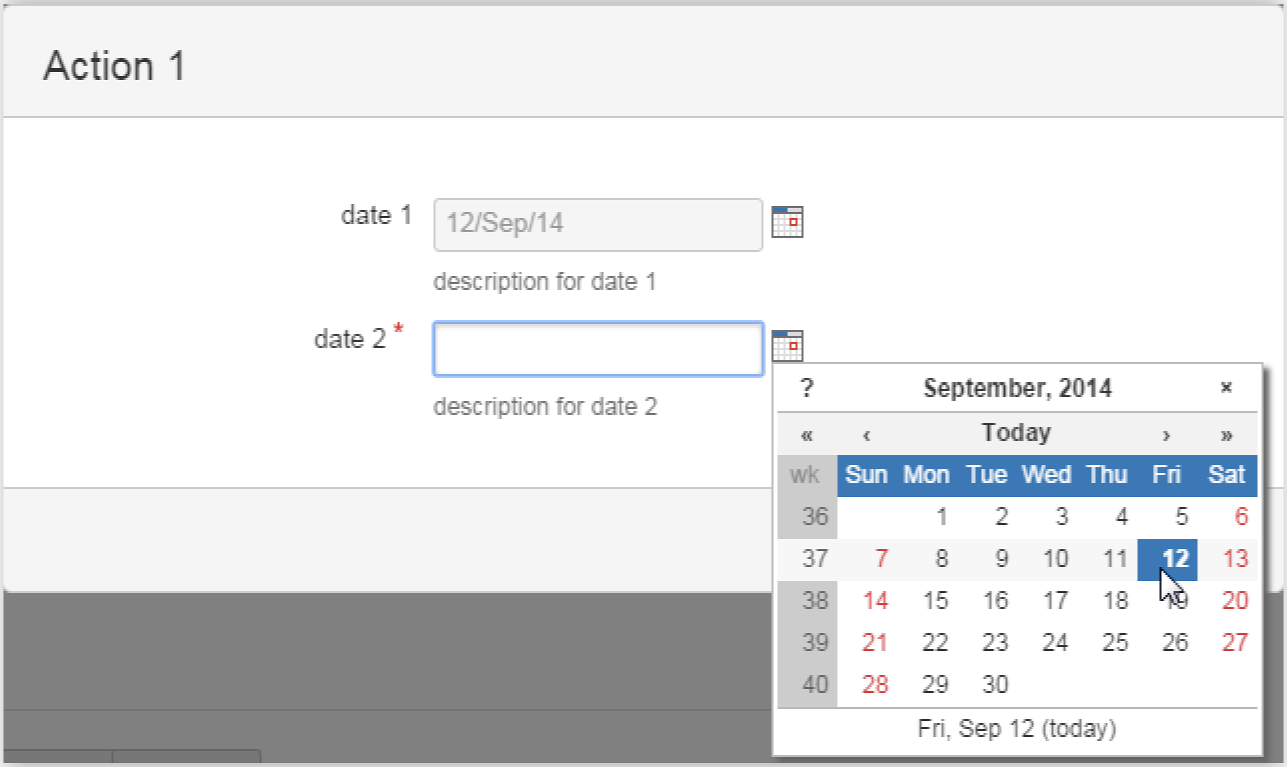
Return type:

The returned values has no meaning

Examples:

Example 1:

```
BA_createDatePicker("date 1", currentDate(), true, false, "description for date 1");
BA_createDatePicker("date 2", (date)"", false, true, "description for date 2");
```



Action 1

date 1: 12/Sep/14

description for date 1

date 2 *

description for date 2

?			September, 2014			×
«	‹		Today		›	»

wk	Sun	Mon	Tue	Wed	Thu	Fri	Sat
36		1	2	3	4	5	6
37	7	8	9	10	11	12	13
38	14	15	16	17	18	19	20
39	21	22	23	24	25	26	27
40	28	29	30				

Fri, Sep 12 (today)

BA_createDateTimePicker

Availability
This routine is available since **Blitz-Actions 2.0.8 and 2.6.1 .**

Syntax:

BA_createDateTimePicker(label, defaultValue, isDisabled, isRequired, fieldDescription)

Description:

Creates a date time picker.

Note that the default value of the date time picker uses the server's timezone. The value shown to the user uses the user's time zone.

Parameters:

Parameter name	Type	Required	Description
label	string	Yes	The label of the field
defaultValue	date	Yes	A default value or an empty string
isDisabled	boolean	Yes	Specifies whether the field is read-only or not
isRequired	boolean	Yes	Specifies if the field is should be marked as required with a dark red asterisk. Note that marking the field as required does not add any validation.
fieldDescription	string	Yes	A description of the field to be displayed immediately under the input box.

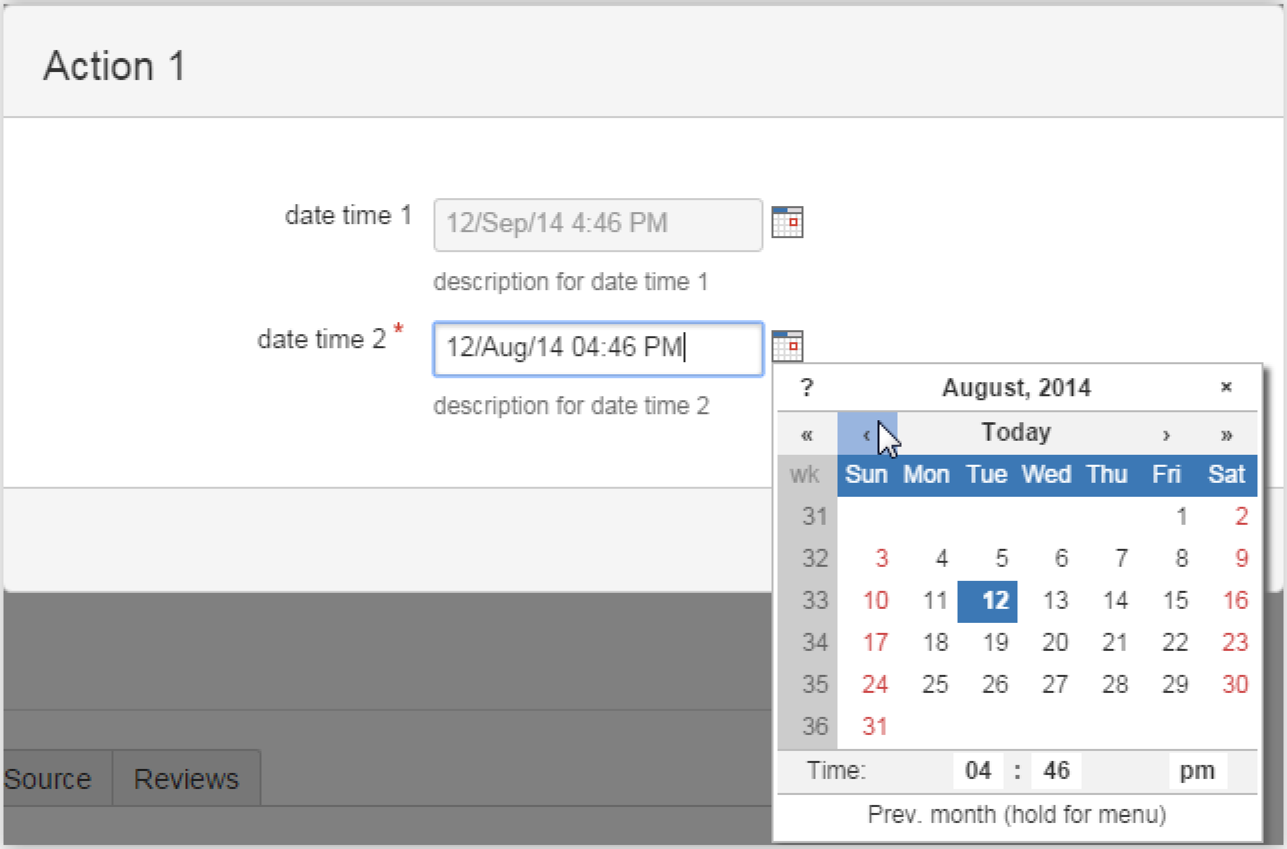
Return type:

The returned values has no meaning

Examples:

Example 1:

```
BA_createDateTimePicker("date time 1", currentDate(), true, false,
"description for date time 1");
BA_createDateTimePicker("date time 2", (date)"", false, true, "description
for date time 2");
```



Action 1

date time 1: 12/Sep/14 4:46 PM

description for date time 1

date time 2 *: 12/Aug/14 04:46 PM

description for date time 2

?	August, 2014						×
«	‹	Today				›	»
wk	Sun	Mon	Tue	Wed	Thu	Fri	Sat
31						1	2
32	3	4	5	6	7	8	9
33	10	11	12	13	14	15	16
34	17	18	19	20	21	22	23
35	24	25	26	27	28	29	30
36	31						

Time: 04 : 46 pm

Prev. month (hold for menu)

BA_createFileUpload

Syntax:

BA_createFileUpload(label, isDisabled)

Since version 2.0.8 and 2.6.1

BA_createFileUpload(label, isDisabled[, isRequired, fieldDescription])

Description:

Creates a file upload field.

Creates an upload control that allows the user to upload multiple files to a temporary folder. See The Administration Page for more details about the temporary file directory.

Parameters:

Parameter name	Type	Required	Description
label	string	Yes	The label of the field
isDisabled	boolean	Yes	Specifies whether the field is read-only or not
isRequired	boolean	No	Specifies if the field is should be marked as required with a dark red asterisk. Note that marking the field as required does not add any validation.
fieldDescription	string	No	A description of the field to be displayed immediately under the input box.

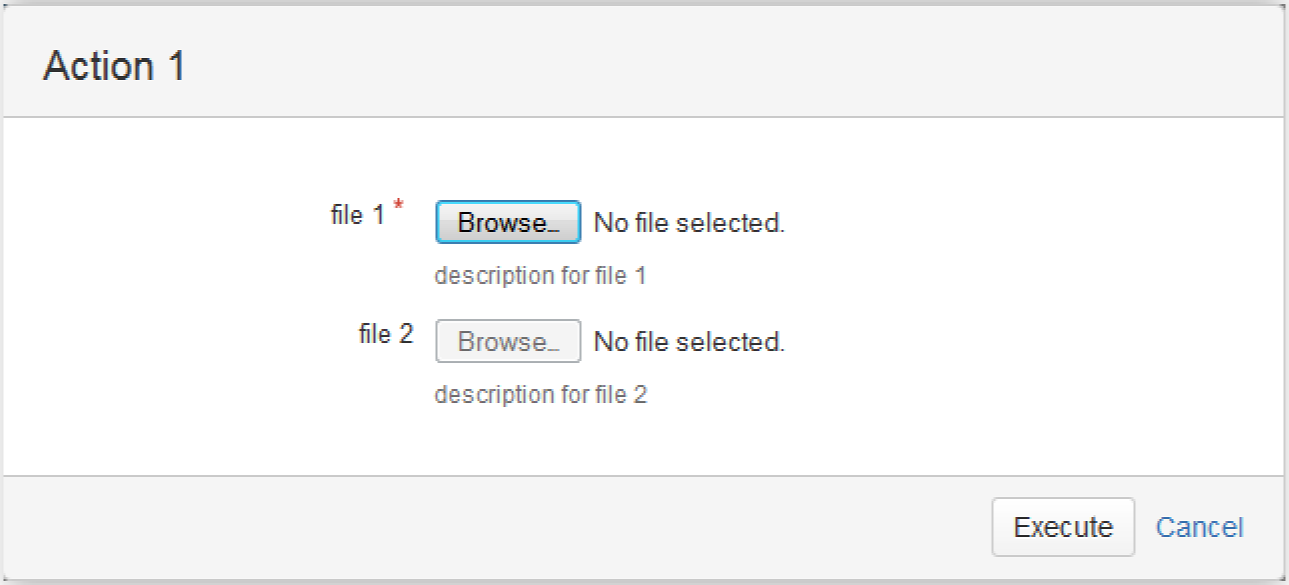
Return type:

The returned values has no meaning

Examples:

Example 1:

```
BA_createFileUpload("file 1", false, true, "description for file 1");
BA_createFileUpload("file 2", true, false, "description for file 2");
```



BA_createHtmlContent

> **Availability**
> This routine is available since **Blitz-Actions 2.0.8 and 2.6.1 .**

Syntax:

BA_createHtmlContent(html);

Description:

Adds plain HTML content to the screen.

Parameters:

Parameter name	Type	Required	Description
html	string	Yes	The HTML content.

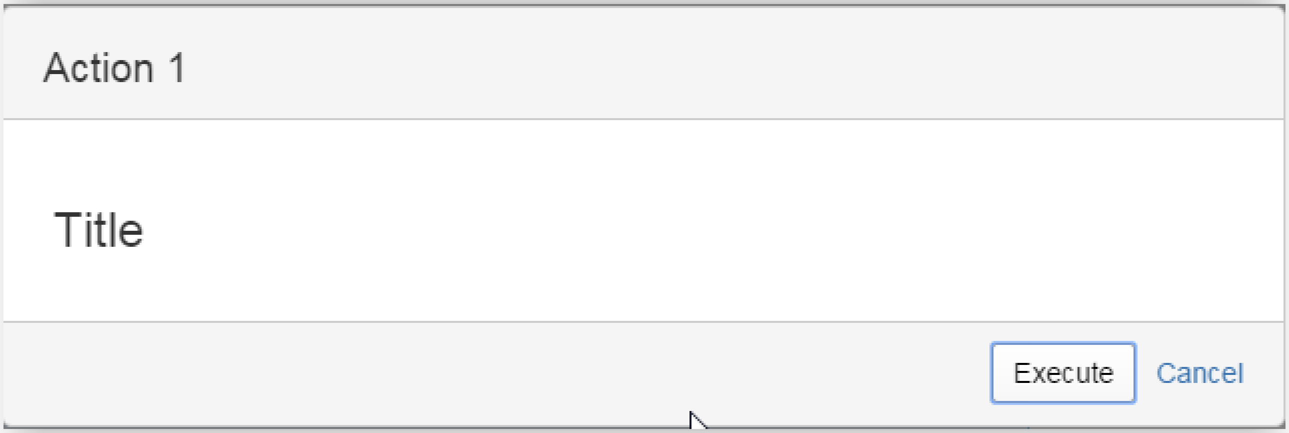
Return type:

The returned value has no meaning.

Examples:

Example 1:

```
BA_createHtmlContent("<h1>Title</h1>");
```



BA_createInput

Syntax:

BA_createInput(label, defaultValue, isDisabled)

Since version 2.0.8 for JIRA 5 and 2.6.1 for JIRA 6:

BA_createInput(label, defaultValue, isDisabled[, isRequired, fieldDescription])

Description:

Creates a simple text input suitable for short values.

Parameters:

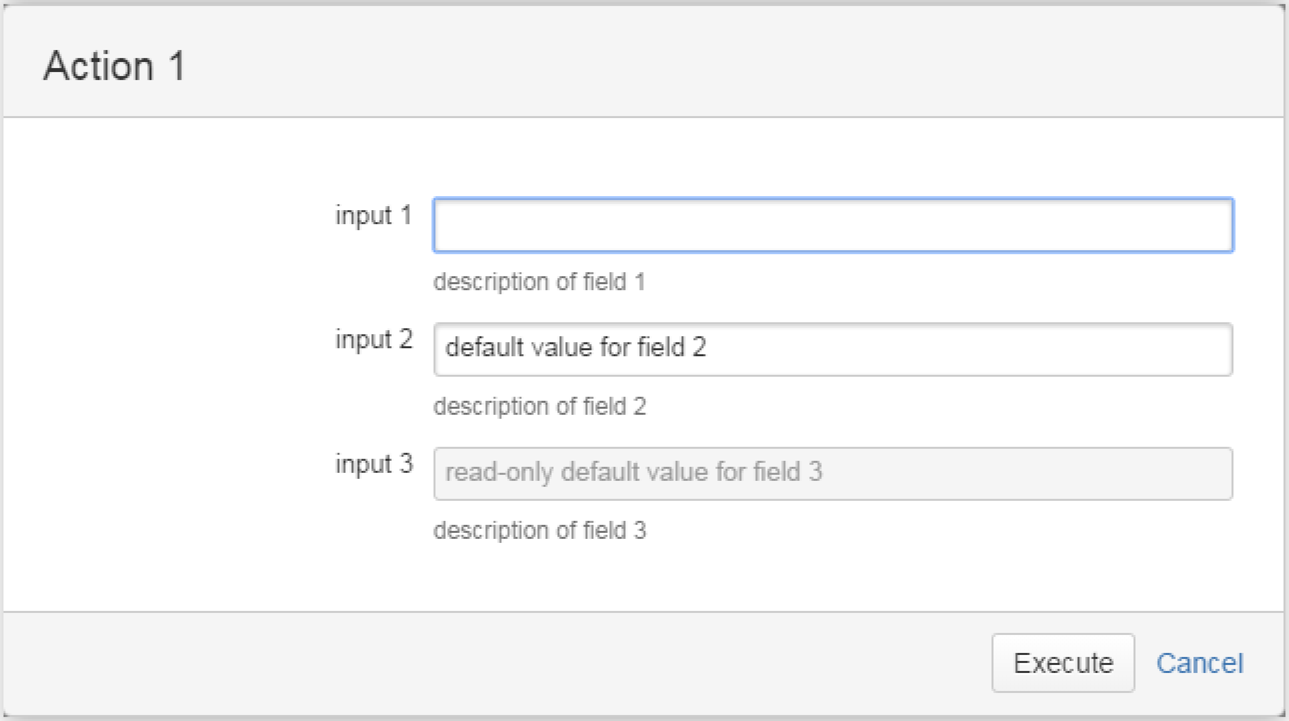
Parameter name	Type	Required	Description
label	string	Yes	The label of the field
defaultValue	string	Yes	A default value or an empty string
isDisabled	boolean	Yes	Specifies whether the field is read-only or not
isRequired	boolean	No	Specifies if the field is should be marked as required with a dark red asterisk. Note that marking the field as required does not add any validation.
fieldDescription	string	No	A description of the field to be displayed immediately under the input box.

Return type:

string []

Example

```
BA_createInput("input 1", "", false, false, "description of field 1");
BA_createInput("input 2", "default value for field 2", false, false,
"description of field 2");
BA_createInput("input 3", "read-only default value for field 3", true,
false, "description of field 3");
```



Action 1

input 1

description of field 1

input 2 default value for field 2

description of field 2

input 3 read-only default value for field 3

description of field 3

BA_createMultiSelectList

Syntax:

BA_createMultiSelectList(label, options, defaultValues, isDisabled)

Since version 2.0.8 and 2.6.1:

BA_createMultiSelectList(label, options, defaultValues, isDisabled[, isRequired, fieldDescription])

Description:

Creates a multi select list.

Parameters:

Parameter name	Type	Required	Description
label	string	Yes	The label of the field
options	string []	Yes	The list of selectable options.
defaultValue	string []	Yes	A sub-list of the options provided or an empty array
isDisabled	boolean	Yes	Specifies whether the field is read-only or not
isRequired	boolean	No	Specifies if the field is should be marked as required with a dark red asterisk. Note that marking the field as required does not add any validation.
fieldDescription	string	No	A description of the field to be displayed immediately under the input box.

Return type:

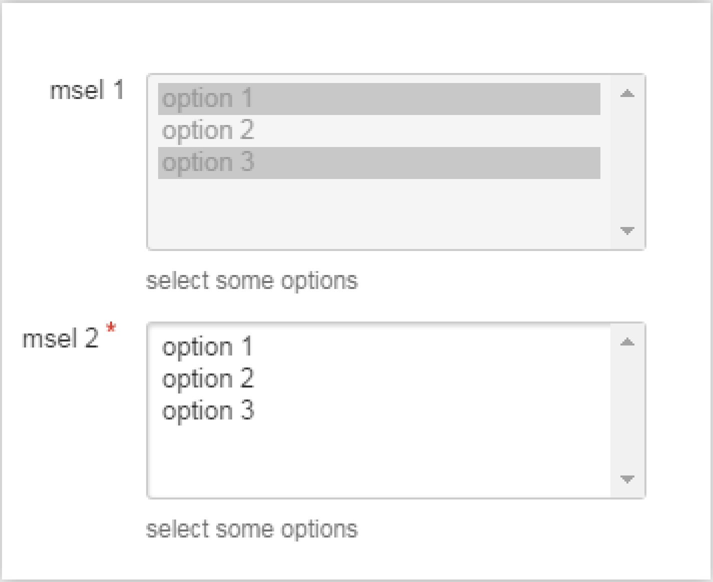
string []

Examples:

Example 1:

```
BA_createMultiSelectList("msel 1", {"option 1", "option 2", "option 3"},
{"option 1", "option 3"}, true, false, "select some options");
BA_createMultiSelectList("msel 2", {"option 1", "option 2", "option 3"},
"", false, true, "select some options");
```

—



BA_createRadioGroup

Syntax:

BA_createRadioGroup(label, options, defaultValue, isDisabled)

Since version 2.0.8 and 2.6.1:

BA_createSelectList(label, options, defaultValue, isDisabled[, isRequired, fieldDescription])

Description:

Creates a radio group.

Parameters:

Parameter name	Type	Required	Description
label	string	Yes	The label of the field
options	string []	Yes	The list of selectable options.
defaultValue	string	Yes	A default value (one of the options provided) or an empty string
isDisabled	boolean	Yes	Specifies whether the field is read-only or not
isRequired	boolean	No	Specifies if the field is should be marked as required with a dark red asterisk. Note that marking the field as required does not add any validation.
fieldDescription	string	No	A description of the field to be displayed immediately under the input box.

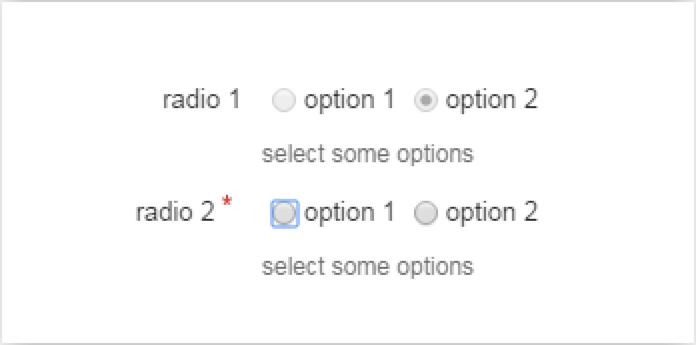
Return type:

string []

Examples:

Example 1:

```
BA_createRadioGroup("radio 1", {"option 1", "option 2"}, "option 2", true,
false, "select some options");
BA_createRadioGroup("radio 2", {"option 1", "option 2"}, "", false, true,
"select some options");
```



radio 1 ○ option 1 ● option 2

select some options

radio 2 * ○ option 1 ○ option 2

select some options

BA_createSelectList

Syntax:

BA_createSelectList(label, options, defaultValue, isDisabled)

Since version 2.0.8 and 2.6.1:

BA_createSelectList(label, options, defaultValue, isDisabled[, isRequired, fieldDescription])

Description:

Creates a single select list.

Parameters:

Parameter name	Type	Required	Description
label	string	Yes	The label of the field
options	string []	Yes	The list of selectable options.
defaultValue	string	Yes	A default value (one of the options provided) or an empty string
isDisabled	boolean	Yes	Specifies whether the field is read-only or not
isRequired	boolean	No	Specifies if the field is should be marked as required with a dark red asterisk. Note that marking the field as required does not add any validation.
fieldDescription	string	No	A description of the field to be displayed immediately under the input box.

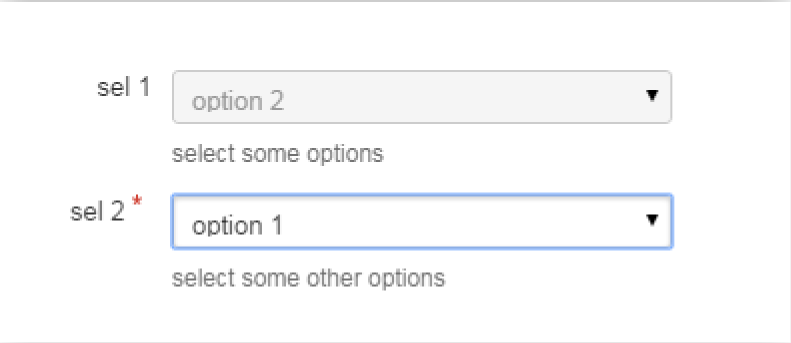
Return type:

string []

Examples:

Example 1:

```
BA_createSelectList("sel 1", {"option 1", "option 2"}, "option 2", true,
false, "select some options");
BA_createSelectList("sel 2", {"option 1", "option 2"}, "", false, true,
"select some other options");
```



sel 1: option 2 ▾

select some options

sel 2 *: option 1 ▾

select some other options

BA_createSingleCheckbox

Syntax:

BA_createSingleCheckbox(label, isChecked, isDisabled)

Since version 2.0.8 and 2.6.1:

BA_createSingleCheckbox(label, isChecked, isDisabled[, isRequired, fieldDescription])n]])

Description:

Creates a single checkbox

Parameters:

Parameter name	Type	Required	Description
label	string	Yes	The label of the field
isChecked	boolean	Yes	The default value
isDisabled	boolean	Yes	Specifies whether the field is read-only or not
isRequired	boolean	No	Specifies if the field is should be marked as required with a dark red asterisk. Note that marking the field as required does not add any validation.
fieldDescription	string	No	A description of the field to be displayed immediately under the input box.

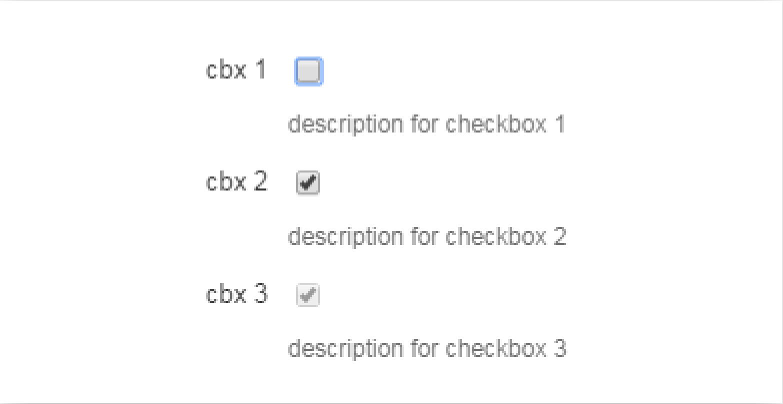
Return type:

string []

Examples:

Example 1:

```
BA_createSingleCheckbox("cbx 1", false, false, false, "description for
checkbox 1");
BA_createSingleCheckbox("cbx 2", true, false, false, "description for
checkbox 2");
BA_createSingleCheckbox("cbx 3", true, true, false, "description for
checkbox 3");
```



cbx 1 ☐

description for checkbox 1

cbx 2 ☑

description for checkbox 2

cbx 3 ☑

description for checkbox 3

BA_createTextArea

Syntax:

BA_createTextArea(label, defaultValue, isDisabled [, rows])

Since version 2.0.8 and 2.6.1:

BA_createTextArea(label, defaultValue, isDisabled [, rows[, isRequired, fieldDescription]])

Description:

Creates a textarea suitable for longer text values like comments.

Parameters:

Parameter name	Type	Required	Description

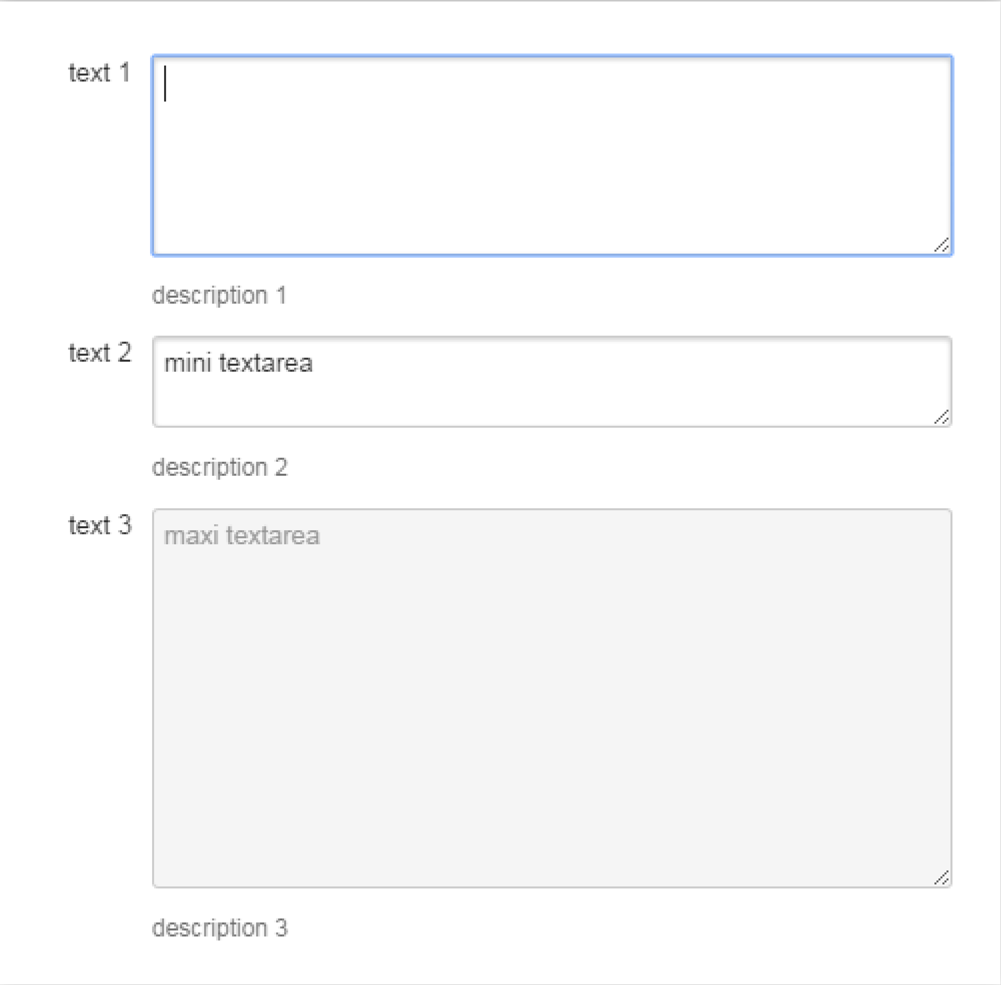
label	string	Yes	Specifies a label for the field
defaultValue	string	Yes	Specifies a default value for the textarea
isDisabled	boolean	Yes	Specifies whether the input should be read-only or not.
rows	number	No	The initial height of the textarea (defaults to 5). Some browsers will allow resizing.
isRequired	boolean	No	Specifies if the field is should be marked as required with a dark red asterisk. Note that marking the field as required does not add any validation.
fieldDescription	string	No	A description of the field to be displayed immediately under the input box.

Return type:

string []

Examples:

```
BA_createTextArea("text 1", "", false , 5, false, "description 1");
BA_createTextArea("text 2", "mini textarea", false , 2, false, "description 2");
BA_createTextArea("text 3", "maxi textarea", true , 10, false, "description 3");
```



text 1

description 1

text 2

mini textarea

description 2

text 3

maxi textarea

description 3

BA_createUserPicker

Availability
This routine is available since **Blitz-Actions 2.0.8 and 2.6.1 .**

Syntax:

BA_createUserPicker(label, defaultValue, isDisabled, isRequired, fieldDescription)

Description:

Creates a user picker.

Parameters:

Parameter name	Type	Required	Description
label	string	Yes	The label of the field
defaultValue	string	Yes	A default value or an empty string. If the default value is not a valid username, it will be discarded and an empty default value will be provided.
isDisabled	boolean	Yes	Specifies whether the field is read-only or not
isRequired	boolean	Yes	Specifies if the field is should be marked as required with a dark red asterisk. Note that marking the field as required does not add any validation.
fieldDescription	string	Yes	A description of the field to be displayed immediately under the input box.

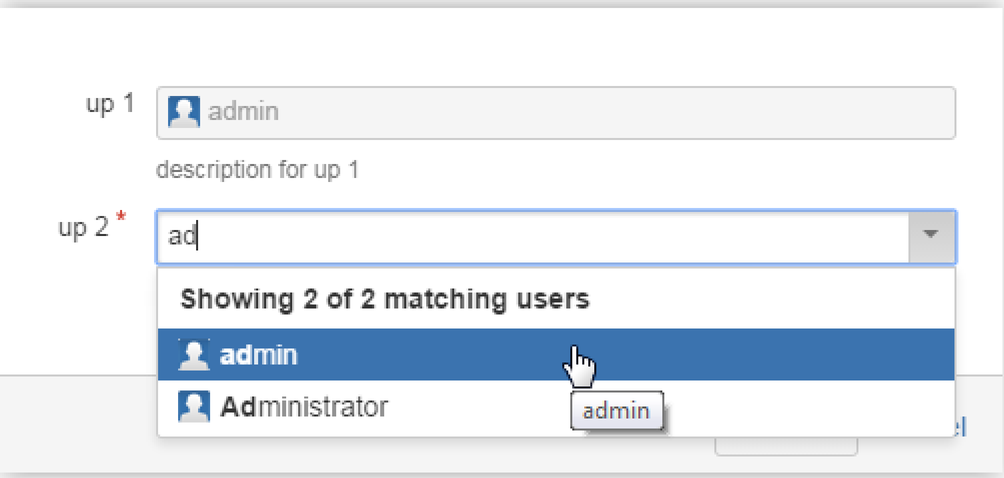
Return type:

The returned values has no meaning

Examples:

Example 1:

```
BA_createUserPicker("up 1", "admin", true, false, "description for up 1");
BA_createUserPicker("up 2", "", false, true, "description for up 2");
```



up 1: admin

description for up 1

up 2 *: ad

BA_createWikiContent

Availability
This routine is available since **Blitz-Actions 2.0.8 and 2.6.1 .**

Syntax:

BA_createWikiContent(wikiText);

Description:

Adds wiki-rendered content to the screen.

Parameters:

Parameter name	Type	Required	Description
wikiText	string	Yes	The wiki content to be rendered content.

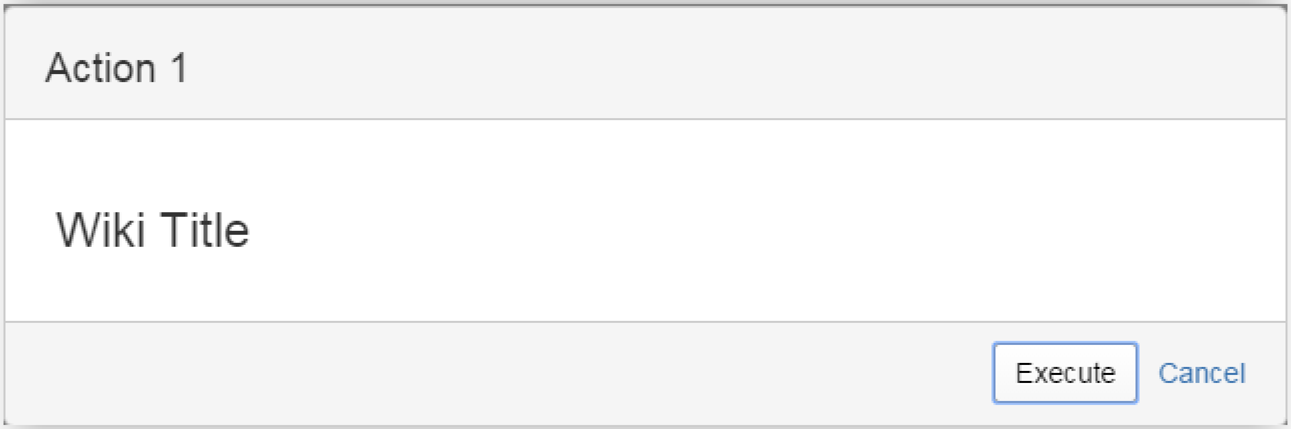
Return type:

The returned value has no meaning.

Examples:

Example 1:

```
BA_createWikiContent("h1. Wiki Title");
```



Action 1

Wiki Title

The Action Script

The Action script contains instructions that will be executed when calling an action. If you're not yet familiar with SIL, read Simple Issue Language Usage.

If the action required some input from the user via the Screen Script, the values are placed in the **argv** variable. This is a string array containing ordered sequences of label and value(s) for the fields shown on the screen.

For single value fields, the array contains the field label immediately followed by the value entered. For multi value fields, if no value was selected, the label is immediately followed by an empty value. If one or more values were selected, the label is immediately followed by the number of values selected, and the selected values.

In order to help you retrieve the selected value(s) faster, we have created more plugin-specific SIL routines.

Category	Used with	Routine	Return value
Single value fields	▪ text ▪ text area ▪ select list ▪ radio group ▪ *single checkbox(optional)*	BA_getSingleValue(argv, label)	String representing the value entered or the selected option for the given label. For checkbox it will return "checked" if the checkbox was selected or an empty string otherwise.

Multi value fields	- multi select list - checkbox group	BA_getMultiValues(argv, label)	String array representing the selected options.
	- file upload (since v. 2.5)		String array containing pairs of original filename and uploaded path. Even indices will contain original filenames, while odd indices will contain uploaded path. See Examples for usage.
Checkbox	- single checkbox	BA_isChecked(argv, label)	Convenience method for checking whether a checkbox is selected or not. Returns a boolean: true if the box was checked, false otherwise.
Date (since v2.0.8 and 2.6.1)	- date picker - date time picker	BA_getDateValue(argv, label)	Will retrieve a date value from the argv array. Note that for the date time picker, the value in the argv variable uses the user's time zone. BA_getDateValue will convert this value from the user's time zone to the server time zone.

These methods are only available in versions 1.0.1+ and 2.0.1+

For example, take the following Screen Script:

```
BA_createCheckboxGroup("checkbox group", {"o1", "o2", "o3"}, {"o1", "o2"},
false);
```

Assuming the user did not change the default values, the **argv** variable in the action script will have 4 values: "checkbox group", "2", "o1", "o2". The result of BA_getMultiValues(argv, "checkbox group") would return an array with two values: o1 and o2.

If the user unchecked all the values, the **argv** variable would have only 2 values: "checkbox group" and an empty string.

You might also use these in conjunction with array-specific routines like arrayGetElement and arrayElementExists. See Array Routines for a list of available routines that manipulate arrays.

Validating the Screen Fields

Once you have a screen that requires additional user input, it is necessary to validate that input and let the user know if s/he did something wrong. This can be done by returning values from the script.

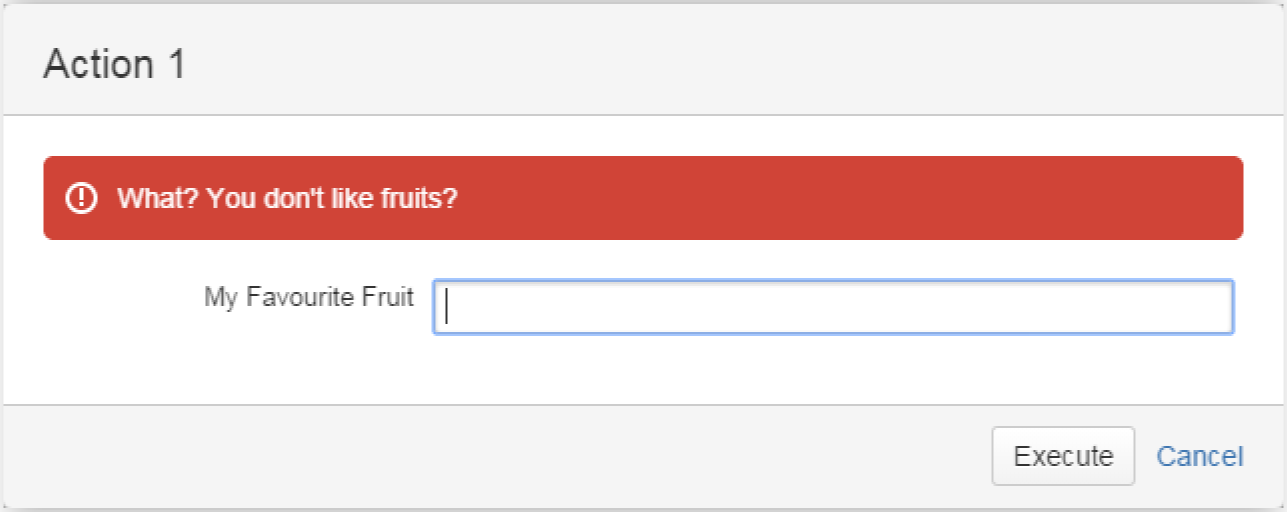
Generic Errors

Generic errors show up in the upper part of the screen, as demonstrated below.

To do this, all you have to do is return the error message. Like this:

```
string fruit = getElement(argv, 1);

if( isNull(fruit) ){
 return "What? You don't like fruits?";
}
```



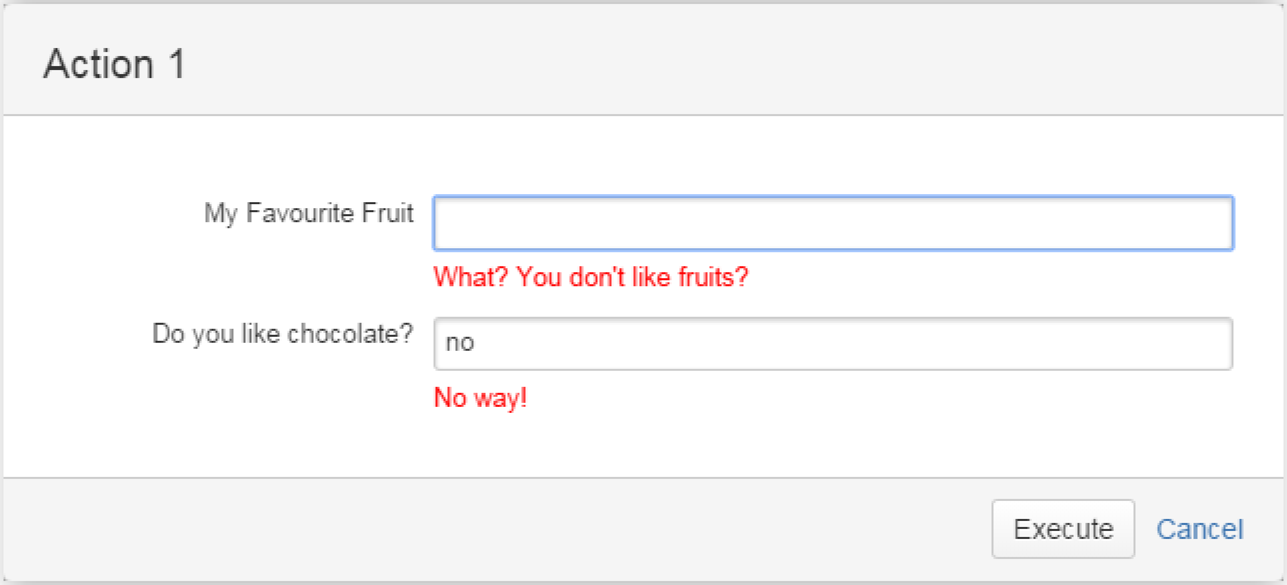
Action 1

What? You don't like fruits?

My Favourite Fruit

Field Validation Errors

If you have multiple input fields on the screen, it would be nice to validate them all at once and show errors for each one.



Action 1

My Favourite Fruit

What? You don't like fruits?

Do you like chocolate? no

No way!

To do this, you will have to return an array containing pairs of label and error.

```
string fruitLabel = getElement(argv, 0);
string fruit = getElement(argv, 1);

string chocolateLabel = getElement(argv, 2);
string chocolate = getElement(argv, 3);

string [] errors;

if( isNull(fruit) ){
  errors = addElement(errors, fruitLabel);
 errors = addElement(errors, "What? You don't like fruits?");
}

if( chocolate != "yes" and chocolate != "of course"){
 errors = addElement(errors, chocolateLabel);
 errors = addElement(errors, "No way!");
}

return errors;
```

Examples

Example 1

Iterating over uploaded files

```
string [] files = BA_getMultiValues(argv, "file");
for(int i = 0; i < size(files); i+=2){
 number ORIGINAL_NAME = i;
 number NEW_PATH = i + 1;
 desc += "File " + files[ORIGINAL_NAME] + " was uploaded to " +
files[NEW_PATH] + "\n";
}
```

_routine_template

Availability
This routine is available since **Blitz-Actions x .**

Syntax:

routine(...)

Since version 2.0.8 and 2.6.1:

BA_createTextArea(label, defaultValue, isDisabled [, rows[, isRequired, fieldDescription]])

Description:

Does something

Parameters:

Parameter name	Type	Required	Description
label	string	Yes	The label of the field
defaultValue	string	Yes	A default value or an empty string
isDisabled	boolean	Yes	Specifies whether the field is read-only or not
isRequired	boolean	No	Specifies if the field is should be marked as required with a dark red asterisk. Note that marking the field as required does not add any validation.
fieldDescription	string	No	A description of the field to be displayed immediately under the input box.

Return type:

string []

Examples:

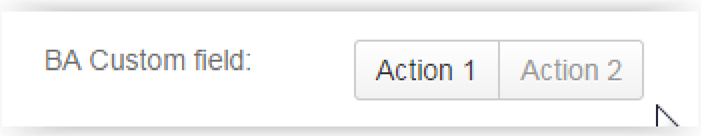
Example 1:

```
wret = chop("Once upon a time", 6);
print(wret);
```

Print ***—Once u***

Using Blitz Actions

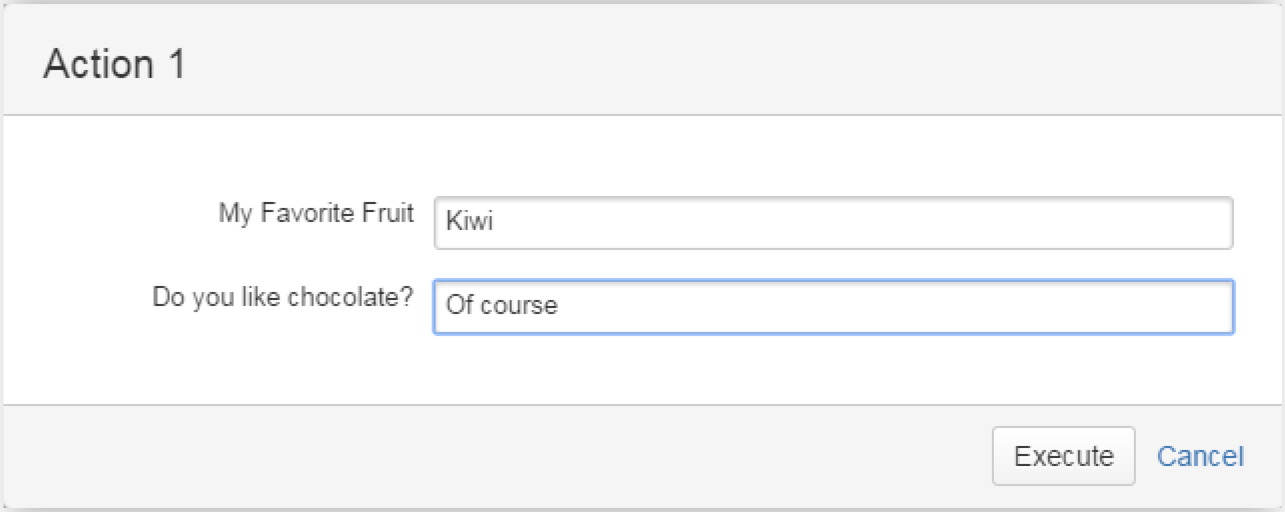
The end-user will see the Blitz Actions in a toolbar on the issue view.



The custom field shown above, actually defines 3 actions, but each has a different condition; *Action 1* is enabled, *Action 2* is disabled and *Action 3* is hidden.

You can easily spot Actions that require a screen by the trailing "..." in the labels.

To execute an action, just click the respective button. If the action requires additional input from the user, a dialog similar to the one below will be shown before executing the Action Script.



Action 1

My Favorite Fruit: Kiwi

Do you like chocolate?: Of course

Info
For more real life examples using Blitz Actions, please visit our Tutorials & Recipes page.

Installation

Installation via Atlassian Universal Plugin Manager

This page points the simple steps to follow for installing the plugin using the Universal Plugin Manager. This method requires an internet connection.

Manual Install

It may seem more complicated, but a manual install is quite easy to do. After all, all you have to do is to copy some files. Here's how.

Good practice
If you are already using a previous version of this plugin and you are installing a new one, please clear the cache of your browser after the installation to avoid errors due to the javascript changes.

Installation via Atlassian Universal Plugin Manager

Installation via Atlassian Universal Plugin Manager

If you are not familiar with Universal Plugin Manager (UPM), please read this document before we begin.

All you have to do is to go to *Administration->Add-ons->Find new add-ons*, search for **Kepler Blitz Actions** and install it.

That's all.

Manual Install

Manual Install

Do not worry, it's a simple task to install it manually:

1. Download the correct Blitz Actions obr file from Atlassian Marketplace or from our site: Kepler Products.

2. Go to Administration->Add-ons->Manage add-ons. Install the previously downloaded obr file by using 'Upload add-on' link.

3. [Optional, but highly recommended]: Enable logging on our modules. Open with a text editor of your choice the JIRA log4j configuration file *JIR*

A_INSTALL_DIR/atlassian-jira/WEB-INF/classes/log4j.properties and add these 2 lines at the end of it. Restart JIRA.

```
log4j.logger.com.keplerrominfo=INFO, filelog
log4j.additivity.com.keplerrominfo=false
```

Licensing

Dual Licensing support

All versions support both Kepler and Atlassian licenses, but you only need one valid license to run the plugin, which can is provided as the **blitzactions.lic** file, or as the key generated via the Atlassian Marketplace.

The order in which the licenses are checked is:

1. Atlassian License
2. Kepler License

It is **strongly recommended** that you do not install both licenses at once, as this might yield unwanted results. For example, consider that you have an Atlassian License with the support date expired and one valid Kepler License. In this case you cannot update the plugin, because the Atlassian License is checked first and its support date is expired.

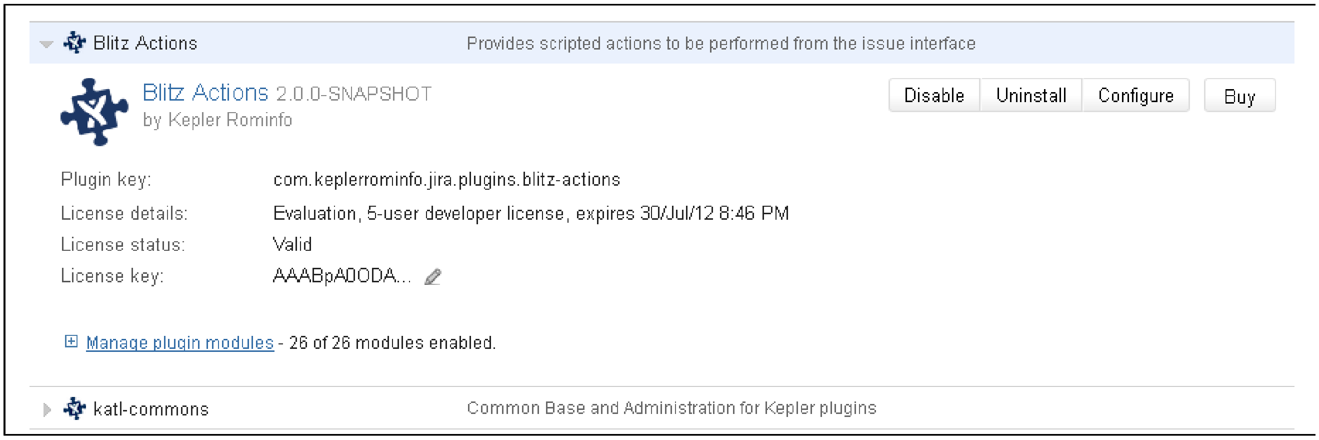
Atlassian Licensing

> To support Atlassian licenses you need to install **katl-commons 2.0.4+** *before* installing Blitz Actions.

The Atlassian Marketplace allows you to easily purchase or generate an evaluation license for **Blitz Actions**.

Using Universal Plugin Manager 2.0.1+

After generating the license key, all you have to do is access the **Administration-> Plugins** section in your JIRA instance and paste the key into the corresponding plugin textbox.



Blitz Actions — Provides scripted actions to be performed from the issue interface

Blitz Actions 2.0.0-SNAPSHOT
by Kepler Rominfo

Plugin key: com.keplerrominfo.jira.plugins.blitz-actions
License details: Evaluation, 5-user developer license, expires 30/Jul/12 8:46 PM
License status: Valid
License key: AAABpA0ODA...

Manage plugin modules - 26 of 26 modules enabled.

katl-commons — Common Base and Administration for Kepler plugins

Kepler Licensing

The Kepler license is a file (**blitzactions.lic**) which must be placed in the <JIRA_HOME>/kepler folder. You can either generate and download a free evaluation license by registering on our site and accessing the **Licenses** section, or you can purchase the plugin by following these instructions.

> **Reminder**
> Don't forget that you need only *one* valid license to run the plugin.

Technical info
Starting with **katl-commons version 2.5.5** an new plugin, called ***Warden***, will be automatically installed by any paid add-on (including Kepler Blitz Actions). This plugin is responsible with the management of licenses (both JIRA and Kepler). Do not attempt to uninstall it without removing first all the Kepler paid add-ons.

Removing an unused license
If you want to remove a no longer used Atlassian license, this can be done in UPM (for UPM 2.0.1+) or by removing the old license key and clicking Update. To remove a Kepler license, you have to delete the correspondent .lic file from the kepler folder. Note that any change to the Kepler license requires a server restart.

Install notes for JIRA 7

When upgrading from an older version of JIRA to JIRA 7, you must update all our plugins as well.

As you can see on this page, the versions compatible with JIRA 7 are the 3.1.x versions.

What should I do if I installed an incompatible version?

As we have said before, **3.0.x** versions are compatible with **JIRA 6.x** and **3.1.x** versions are compatible with **JIRA 7.x**.

If you have installed Blitz Actions 3.0.x on JIRA 7.x or Blitz Actions 3.1.x on JIRA 6.x, you should do the next steps :

1. Uninstall warden
2. Uninstall katl-commons
3. Uninstall Blitz Actions
4. Install the right version of Blitz Actions (the one compatible with your JIRA)
5. katl-commons and warden should now have the right versions as well

After you uninstall katl-commons and warden, some plugins may remain disabled, so you may need to re-enable them manually.

Uninstall

Uninstall via Atlassian Universal Plugin Manager

This page shows the steps to uninstall the plugin using the Universal Plugin Manager.

Manual Uninstall

At first sight, this seem a little bit complicated but actually it isn't. All it has to be done is to remove the plugin manually and delete its tables from the internal database.

- Manual Uninstall
- Uninstall via Atlassian Universal Plugin Manager

Manual Uninstall

Uninstall manually

At first we will uninstall the plugin manually and finally we'll remove the corresponding tables in the internal database.

Uninstall the plugin

You need to have access where the Jira server has been installed.

Goto the folder where Jira server has been installed.

Access **<JIRA_APPLICATION_DATA>/plugins/installed-plugins** and manually delete `Blitz Actions` plugin.

Remove the tables

You can go to the internal database administration tool.

You can use a visual tool or a command line tool and remove the following table in your database:

- kblitzactions

Restart the server

Now you can restart Jira server

Uninstall via Atlassian Universal Plugin Manager

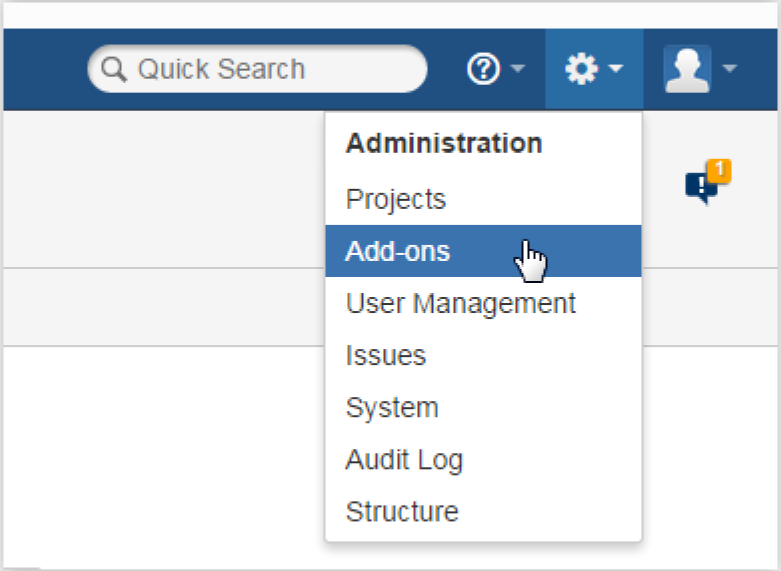
Uninstall via Atlassian Universal Plugin Manager

At first we will uninstall the plugin and finally we'll remove the corresponding tables in the internal database.

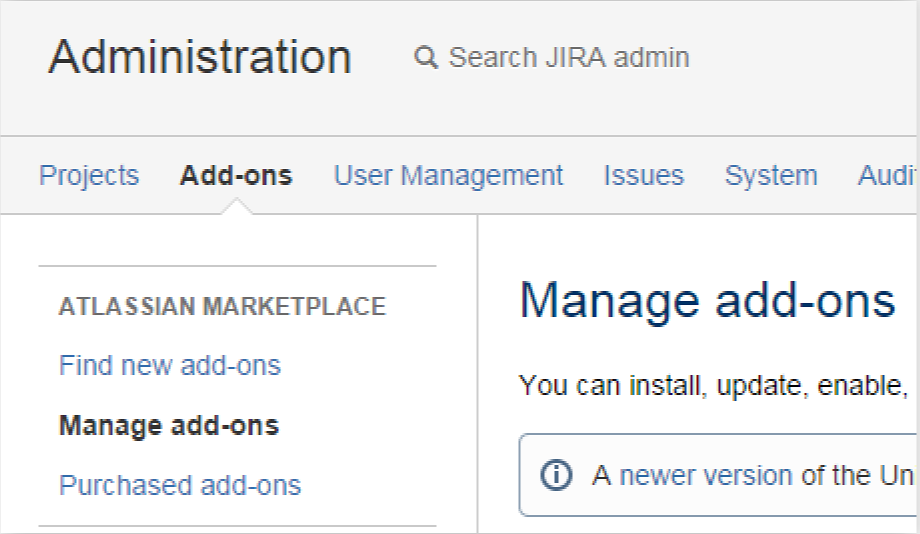
Uninstall the plugin

If you are not familiar with Universal Plugin Manager (UPM), please read this document before we begin.

1) Log in as administrator and go to Jira Administration in the right up section of the page



2) Click on the `Manage Add-ons` menu link from the Add-ons menu section

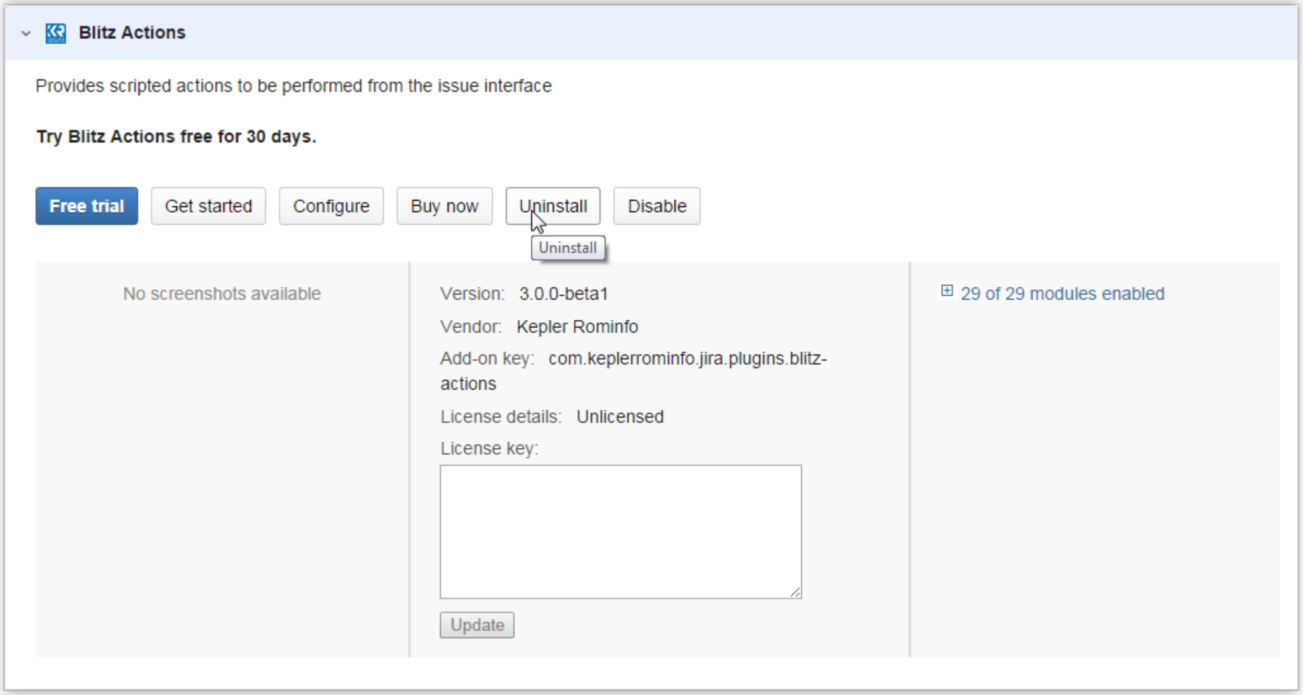


Manage add-ons

You can install, update, enable,

A newer version of the Un

3) Search for `Blitz Actions` plugin in `User Installed Plugins` section and click on `Uninstall` button



Provides scripted actions to be performed from the issue interface

Try Blitz Actions free for 30 days.

Version: 3.0.0-beta1

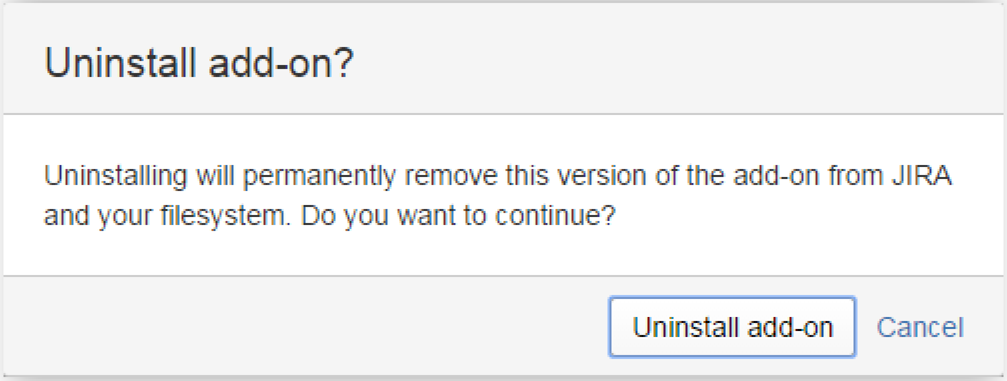
Vendor: Kepler Rominfo

Add-on key: com.keplerrominfo.jira.plugins.blitz-actions

License details: Unlicensed

License key:

4) Press `Continue` when the uninstall confirmation dialog box appears

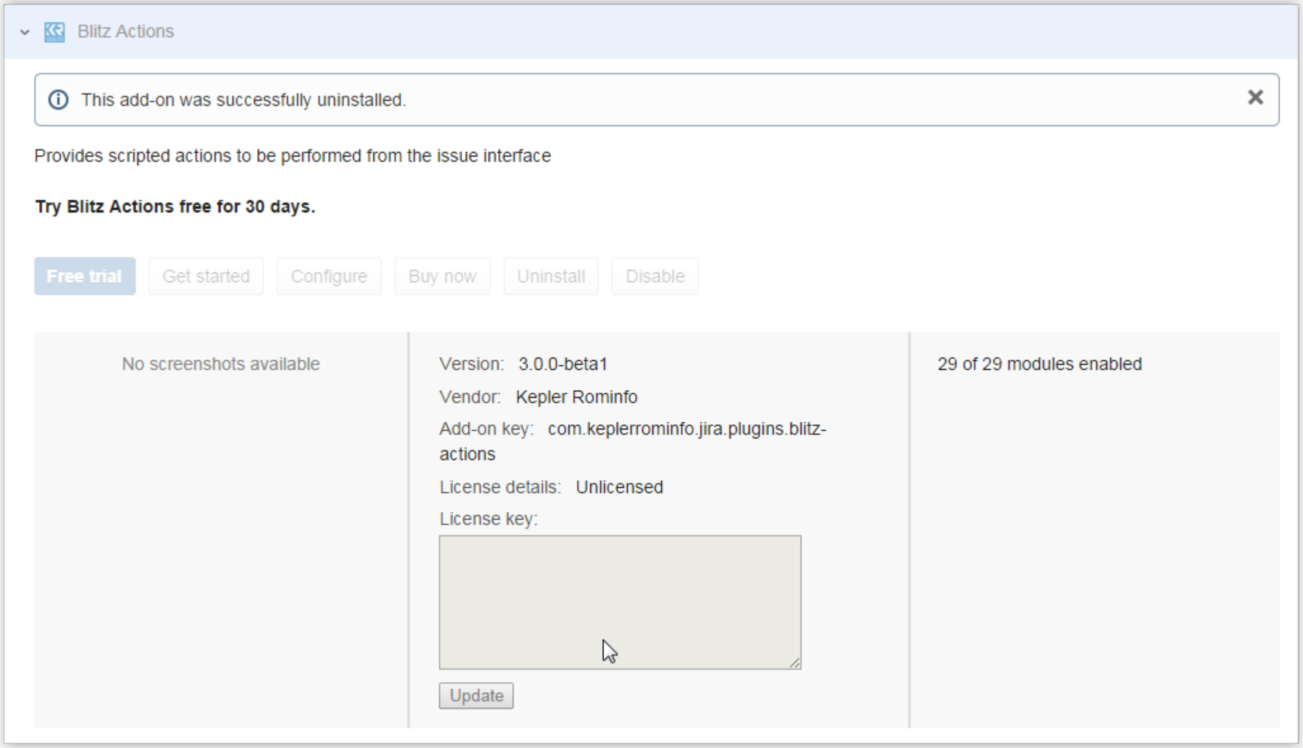


Uninstall add-on?

Uninstalling will permanently remove this version of the add-on from JIRA and your filesystem. Do you want to continue?

Optionally, you can provide feedback on the reason why you uninstalled the plugin.

5) A message "successfully uninstalled" should appear



This add-on was successfully uninstalled.

Provides scripted actions to be performed from the issue interface

Try Blitz Actions free for 30 days.

No screenshots available

Version: 3.0.0-beta1

Vendor: Kepler Rominfo

Add-on key: com.keplerrominfo.jira.plugins.blitz-actions

License details: Unlicensed

License key:

29 of 29 modules enabled

Now you can delete Blitz Actions corresponding plugin tables.

Remove the tables

You can go to the internal database administration tool.

You can use a visual tool or a command line tool and remove the following table in your database:

- kblitzactions

Backup and restore

At Restore: install first the plugins

Mundane operations as backup and restore may pose some problems to the unsuspecting JIRA administrator. Since all the Kepler plugins create some tables in the JIRA schema - we created this mechanism long before Active Objects was introduced into Atlassian's framework - you need to take some precautions at restore.

Specifically, at restore you need to create the tables used by our plugins. You do not need to copy schema from the previous JIRA or fill it with data, you just need to **simply install the plugins into JIRA before restoring** (enabling the plugins would create the needed tables).

Blitz Actions has two dependencies:

1. katl-commons (core support)
2. warden (used for licensing)

For reference, these are the tables created for each add-on

Plugin	Tables

Blitz Actions	kblitzactions
katl-commons	kplugins kpluginscfg kissuestate kstatevalues
warden	-

Requirements

A fully installed Blitz Actions consists of multiple jar files. You are advised to use the bundle installer when installing Blitz Actions. Please refer to the Install Guide for explanations and details.

At the minimal level Blitz Actions consists from 2 dependencies (jar files): katl-commons (a library having countless utility routines, but also - most important - the SIL language parser) and Blitz Actions jar file, which contains Blitz Actions specific routines plus the user interface (e.g.: script editors)

Blitz Actions	JIRA	katl-commons
3.0.0	6.x	3.0.0
3.0.1	6.x	3.0.1
3.0.2	6.x	3.0.2
3.0.3	6.x	3.0.3
3.0.4	6.x	3.0.4
3.0.5	6.x	3.0.6
3.0.6	6.x	3.0.10
3.1.0	7.x	3.1.1