Workflow Designer

Creating Tasks

This Page describe a feature, which require PHP development skills. If you use this feature in a wrong way, you could crash Workflows!

Create stub

To create a new custom task you only need to go to the “Task Management” and press “create new block manually” at bottom of the page.

The Workflow Designer will ask you some questions, how you would define the new task.

1. The Key

This is the slug of the Block, which will be used to store the relation in the database.
Must be unique and must not contain spaces or special characters!

2. The Classname

Must be a value with the Prefix “WfTask”. For example WfTaskNewBlock. This is the classname of the new block.

3. The label

The label, visible in the sidebar and below every block in Designer. Should describe the function of this block.

Then the Workflow Designer will create the necessary files.
The block will be created in the task group “special tools” at first.

Now you need to modify the files.

Files

WfTask*.php

You found this files in **/modules/Workflow2/tasks/**.
This file must provide the function to process this action.

```
<?php
require_once(realpath(__DIR__ . '/../autoload_wf.php'));

class dummyTypeClass extends \Workflow\Task
{
```



```
    public function init() {
        // Do some initialize tasks
    }

    public function handleTask($context) {
        /* Insert here source code to execute the task */

        return "yes";
    }

    public function beforeGetTaskform($viewer) {
        /* Insert here source code to create custom configurations pages
*/
    }
    public function beforeSave(&$values) {
        /* Insert here source code to modify the values the user submit
on configuration */
    }
}
```

The require_once in second line will make sure all Workflow related functions are available.

There is an easier way to create configuration. See "User SimpleConfig"

handleTask($context)

This function will run every time the task will be executed and must include all php code you want to run in this situation.

This function **MUST** have a return value, which is equal to one of the output points you configure in task.xml or "*yes*" if created manually in Task Management.

The parameter is from the Type **\Workflow\VTEntity**.

beforeGetTaskform($viewer)

This function will be called before the configuration popup is shown.
You could prepare all dependencies and assign values to the $viewer parameter, which is the Smarty Object used for the popup.

beforeSave($values)

This function will be called before the configuration values are saved.

The $values variable is an array with the submitted fields in the task variable.

WfTask*.js

You found in **/modules/Workflow2/tasks/**.

Will be loaded in the configuration popup and could handle all javascript functions for this window.

WfTask*.tpl

You found in **/layouts/v7/modules/Settings/Workflow2/tasksforms/**

The wildcard in filename will be replaced by the key of task and not the classname, like before.

There exists an open <form Tag, which will store any value you write to the**task[...]** variable and give this value also back to the templates **$task** variable.

Use SimpleConfig

SimpleConfig is a Preset to provide a very easy way to create a task configuration.

You should create configuration within **init()** Function of your custom task, like this:

```
//....
    public function init()
    {
        $this->_SC = $this->addPreset("SimpleConfig", "details", array(
            'templatename' => 'templatevariable',
        ));

                // Check if task is currently in Configuration mode.
Otherwise you don't need to add configuration fields
        if($this->isConfiguration()) {
                        // How much columns your configuration have (Must
be static and set as first)
            $this->_SC->setColumnCount(1);

                        // Add first field to configuration
            $this->_SC->addFields('variablename', 'Configuration label',
'type', array(
                                // ... additional configurations
                                // 'description' => 'Short description of
```



```
field'
            ));

                    $this->_SC->addFields('roomid', 'Into Room',
'select', array(
               'description' => 'Short description of field'
            ));

            $this->_SC->addFields('roomid', 'Into Room', 'select', array(
               'description' => 'Short description of field'
            ));

        }

    }


// ...
```

To access this variables within handleTask, there exist a simple function, shown in this example

```
// ...
  public function handleTask(&$context) {

      if($this->_SC->has('providerid')) {
          $providerid = $this->_SC->get('providerid');

                  $roomid = $this->_SC->get("roomid");
      }

          return "yes";
  }

// ...
```

Available fieldtypes

The list of available field types is an interface, you can extend easily.

Types, the Workflow Designer core provide:

userpicklist	Allow to select a user

password	masked Passwort input
hidden	Hidden value
text	Simple Text input, without template functions
template	Simple Text input, with template function
textarea	Textarea with template functions
expressionfield	Field with custom Expression function
expressionarea	Textarea with custom Expression functions
select,picklist	Single Picklist
multiselect,multipicklist	Multi Picklist
fields	Selection of a Field
checkbox	Checkbox
timezone	Timezone selection
provider	Provider selection
readonly	Readonly text

