# Workflow Designer

## custom Frontend Integration Type


*Copyright © 2022 Redoo Networks GmbH*

You are able to define a new Frontend Type, like “Detailview Top Buttons”, “More Action Button”, …

You can use this to implement Workflow Designer within your module and give the configuration, which workflow should be shown, over “Frontend Manager” configuration panel to the user.

## Register your new Frontend Type

Use the following PHP code during module setup to create a new Type in Frontend Manager.

```
$type = new \Workflow2_FrontendType_Model();
$type->setKey('type-id');

// Colorpicker
$type->addConfig('fieldname1', 'Button1 Color',
\Workflow2_FrontendType_Model::TYPE_COLORPICKER, '#373573');
// Checkbox
$type->addConfig('fieldname2', 'Button1 Color',
\Workflow2_FrontendType_Model::TYPE_CHECKBOX, '1');
// Field selection
$type->addConfig('fieldname3', 'Button1 Color',
\Workflow2_FrontendType_Model::TYPE_FIELDDSELECT);

// Name of your Frontend Type - User will see this within selection
$type->setTitle('Label of this new Type');
// Define your ModuleName, which will be used for translations in
configuration
$type->setRelatedName('ModuleName');
// You are able to define
$type->setEnvironmentHandler('ClassName', 'relative filepath to class');

// save type permanently
$type->save();
```

Everytime you execute this code, the Type is updated. So you are able to modify your Type.

| Parameter | Description |
|---|---|
| type-id | This must be a unique key, which you use to retrieve Buttons/Configurations in Frontend |
| Label of type | This title is shown within Frontend Manager |
| Related Modulename | This module is used for translations of Config Label and Type label |


*Copyright © 2026 Redoo Networks GmbH*

| | |
|---|---|
| EnvironmentHandler / Classname | This class must extend Workflow2_EnvironmentHandlerAbstract_Model and will be called with environment you send during Workflow Trigger and the CRMID of current Record |
| EnvironmentHandler / Filepath | Path, relative to VtigerCRM Root folder, which contain the EnvironmentHandler – Classname |

You can add config values, the user can define per button / link.

First parameter is “config key“, you also get in Frontend
Second parameter is label, shown in Frontend Manager.
This parameter is the Type. Available are “Colorpicker”, “Checkbox” and “Fieldselection”.

Available field types are:

| class constant | Type |
|---|---|
| \Workflow2_FrontendType_Model::TYPE_CHECKBOX | Checkbox |
| \Workflow2_FrontendType_Model::TYPE_COLORPICKER | Colorpicker |
| \Workflow2_FrontendType_Model::TYPE_FIELDDSELECT | Multiple fileselection |

The last parameter is the default value of this config value.

## Example EnvironmentHandler

You can define an extra Environment Loader to load additional data into $env variable of the Workflow, which is triggered from Frontend. The returned array will be added to $env Variable of every Workflow, which is triggered.

```
class RedooPhoneCalls_EnvironmentHandler_Model extends
Workflow2_EnvironmentHandlerAbstract_Model
{

    public function retrieve($environment, $crmid)
    {
                $setEnvironment = array();


                // Load custom data from external sources
                // Process values you get from frontend to build easier
workflows
                // Execute any PHP Code you want to execute from Frontend
Trigger
```

 *Copyright © 2026 Redoo Networks GmbH*

```
                $setEnvironment["dummyValue1"] = "nothing1";
                $setEnvironment["dummyValue2"] = "nothing2";


                return $setEnvironment;
    }
}
```

## Retrieve configurations

Javascript

To retrieve the frontend configurations, you can use the following javascript example:

```
WorkflowFrontendTypes.getWorkflows('type-id', ['recordmodules',
[recordid]]).then(function(response) {
    var buttons = [];

    jQuery.each(response, function(index, value) {
        buttons.push('<button class="btn WorkflowButton" data-crmid="' +
CRMID + '" data-workflowid="' + value.workflow_id + '"
style="background-color:' + value.color + ';color:' + value.textcolor +
';">' + value.label + '</button>');
    });

    var buttonHTML = buttons.join(' ');

    // Do something with buttonHTML

    jQuery('.WorkflowButton').on('click', function(e) {
        var target = jQuery(e.currentTarget);

        var workflowId = target.data('workflowid');
        var targetCRMID = target.data('crmid');

        WorkflowFrontendTypes.trigggerWorkflow('type-id', workflowId,
targetCRMID, { /* Custom Environment */ });
    });
});
```

The call of getWorkflows function have an optional parameter “recordid”. When you set this parameter, the button list is loaded by ajax and takes the start condition into account. The button is

 *Copyright © 2026 Redoo Networks GmbH*

not returned, when the condition is false.
When you don’t define this parameter, you get all frontend buttons, with no guarantee, the workflow really can be executed for a record. But without this parameter, there is no ajax request to server, which provide a much better performance, because it is saved within JavaScript file.

The type-id parameter ist the type key of your Frontend Type, you registered by previous script.

The optional parameter moduleName filter the buttons you get by module. You are completely free, how you show the buttons.

The execution of the Workflow is done by the line:

```
Promise = WorkflowFrontendTypes.trigggerWorkflow('type-id', workflow_id,
target_crmid, { "environment":"value" });
```

| Parameter | Description |
|---|---|
| type-id | Type Key of your type |
| workflow-id | Workflow ID you want to execute |
| crmid | CRMID, which will take as Context of Workflow and where variables are loaded from |
| environmental array | Additional array, which you can access from $env variable in Workflow |

This function will return a jQuery Promise to give you a chance trigger something, when Workflow is finished. Also you will get the complete response array, to get access to javascript response data you set during Workflow. The default Workflow handling of reloading page won’t be applied.

Example of return Promise

```
WorkflowFrontendTypes.triggerWorkflow('type-id', workflow_id,
target_crmid, { "environment":"value" }).then(function(response) {
    // do something when workflow is completed
    // response.responsedata will the the array you set during
"javascript" response block
});
```

PHP

You also can load the Links directly in PHP. There you need to call:

```
$links = \Workflow2_FrontendManager_Model::getLinks("type-id",
```

 *Copyright © 2026 Redoo Networks GmbH*

```
["modulename", ["crmid"]]);
```

Also here type-id is your Type Key, modulename can be used to filter results. When crmid is set the start conditions directly are checked and you will only get valid buttons for this record.


*Copyright © 2026 Redoo Networks GmbH*

*Copyright © 2026 Redoo Networks GmbH*