

# Event actions

- [Overview](#)
- [Create new event action](#)
- [Update existing event action](#)
- [Delete event action](#)

# Overview

**Event actions** are automated actions, performed by the system, when a set of conditions is met or a certain event is triggered (updating of a product, changing an order's status, creating a new invoice...).

Event actions

Search...

+

<input type="checkbox"/>	NAME ^	FAIL ON ERROR	NOTIFY ON ERROR	ENABLED	ACTIONS
<input type="checkbox"/>	Change status when awaited product is booked to stock	✗	✓	✓	<div><div></div><div></div></div>
<input type="checkbox"/>	Change work order CNC status to "in process"	✓	✓	✓	<div><div></div><div></div></div>
<input type="checkbox"/>	Change work order quality control statud when fully reserved	✓	✓	✗	<div><div></div><div></div></div>
<input type="checkbox"/>	Close QC Tasks on ready for delivery	✗	✓	✓	<div><div></div><div></div></div>
<input type="checkbox"/>	Create birokrat partner	✓	✓	✓	<div><div></div><div></div></div>
<input type="checkbox"/>	Create birokrat product	✓	✓	✗	<div><div></div><div></div></div>
<input type="checkbox"/>	Create end customer partner send mail	✓	✓	✓	<div><div></div><div></div></div>
<input type="checkbox"/>	Delete birokrat partner	✓	✓	✓	<div><div></div><div></div></div>
<input type="checkbox"/>	EUC check	✓	✗	✓	<div><div></div><div></div></div>
<input type="checkbox"/>	EUC check sales order	✓	✗	✓	<div><div></div><div></div></div>
<input type="checkbox"/>	End customer check	✓	✗	✓	<div><div></div><div></div></div>
<input type="checkbox"/>	End customer check sales order	✓	✗	✓	<div><div></div><div></div></div>

Event actions are used for a variety of automated tasks, such as sending email notifications, updating statuses, performing hook calls...

Event actions are managed by ERP administrators.

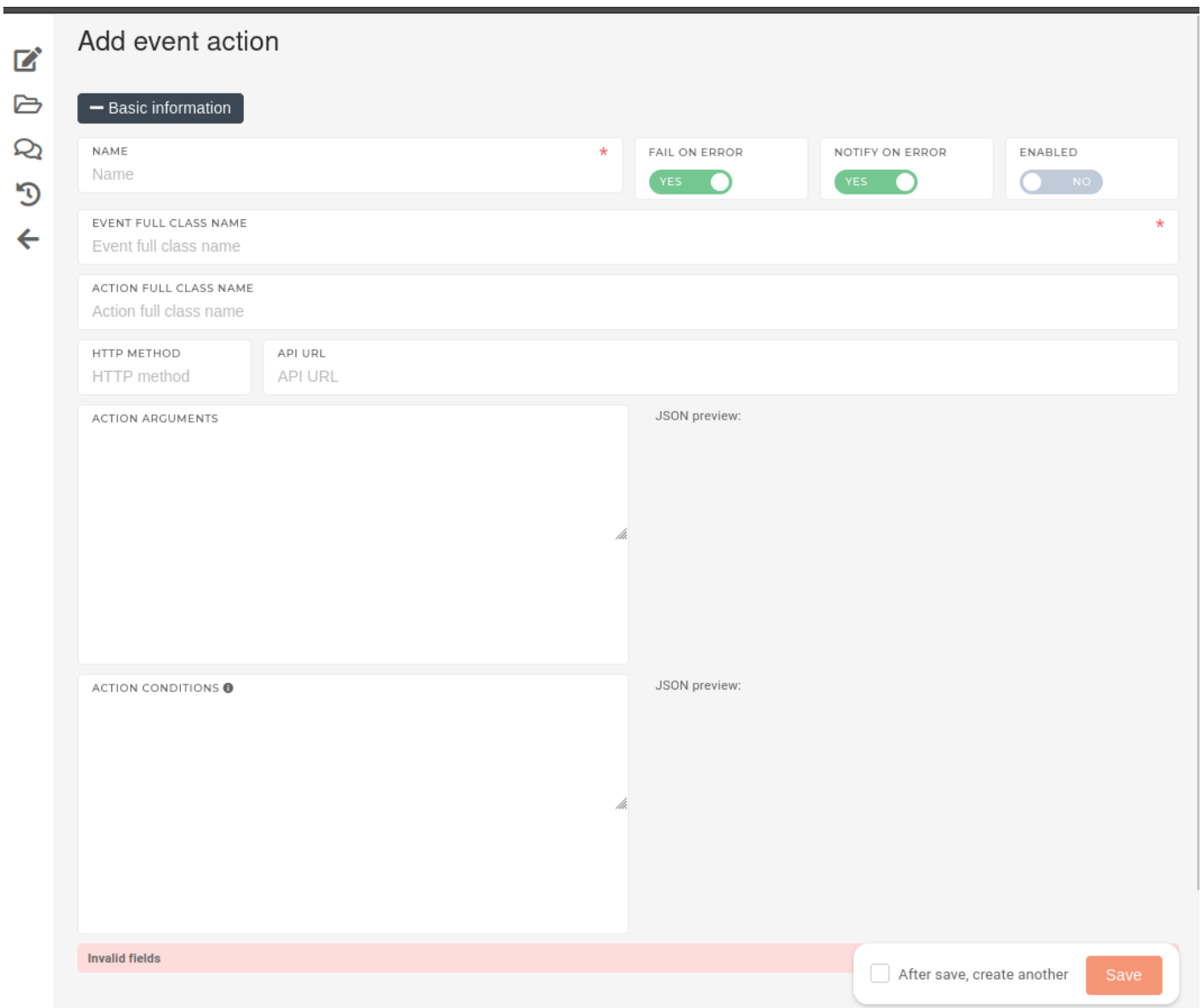
# Create new event action

This action requires the "**create event action**" permission.

To add a new event action, navigate to the "**Add new action**" form, by pressing the "+" button in the top right corner above the table.

Event actions					Q Search...	+	X		
<input type="checkbox"/>	NAME ^	FAIL ON ERROR	NOTIFY ON ERROR	ENABLED	ACTIONS				
<input type="checkbox"/>	Change status when awaited product is booked to stock	✗	✓	✓					
<input type="checkbox"/>	Change work order CNC status to "in process"	✓	✓	✓					
<input type="checkbox"/>	Change work order quality control status when fully reserved	✓	✓	✗					
<input type="checkbox"/>	Close QC Tasks on ready for delivery	✗	✓	✓					
<input type="checkbox"/>	Create biokrat partner	✓	✓	✗					
<input type="checkbox"/>	Create biokrat product	✓	✓	✗					
<input type="checkbox"/>	Create end customer partner send mail	✓	✓	✓					
<input type="checkbox"/>	Delete biokrat partner	✓	✓	✗					
<input type="checkbox"/>	EUC check	✓	✗	✓					
<input type="checkbox"/>	EUC check sales order	✓	✗	✓					
<input type="checkbox"/>	End customer check	✓	✗	✓					
<input type="checkbox"/>	End customer check sales order	✓	✗	✓					
<input type="checkbox"/>	Notify L-TEK when wanted date is changed	✗	✓	✓					
<input type="checkbox"/>	Notify invoicing when pickup task closed	✗	✓	✓					
<input type="checkbox"/>	Notify logistic manager when product without serials created	✗	✓	✓					
<input type="checkbox"/>	Notify logistic manager when product without serials created	✗	✓	✓					
<input type="checkbox"/>	Notify production manager when confirmed date is changed on L-TEK purchase orders	✗	✓	✓					
<input type="checkbox"/>	Notify project owner when note added	✗	✓	✓					
<input type="checkbox"/>	Notify project owner when status changed	✗	✓	✓					
<input type="checkbox"/>	Prevent creating license if DWL and Ulyssix	✓	✓	✗					
<input type="checkbox"/>	Prevent development status change if no file on product	✓	✗	✓					
<input type="checkbox"/>	Product status changed	✓	✓	✓					
<input type="checkbox"/>	Sales price to low notification	✓	✓	✓					
<input type="checkbox"/>	Send email to import-export when sales								

Event actions should only be added by administrators with access to the code base, due to class names and their full paths being mandatory when creating new event actions. That information is only available to developers.



The image shows a web form titled "Add event action". On the left side, there is a vertical toolbar with icons for editing, saving, undo, redo, and a back arrow. The form has a tabbed interface with the "Basic information" tab selected. The form fields include:

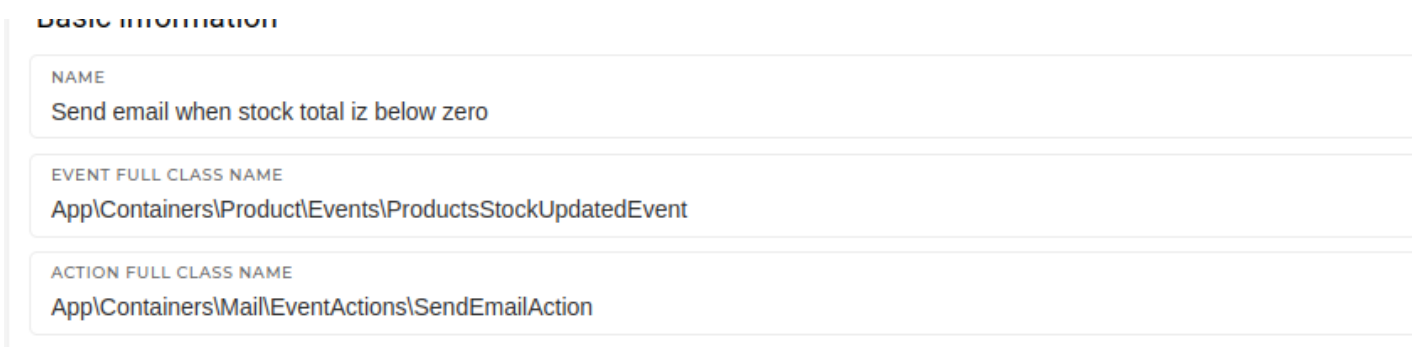
- NAME:** A text input field with a red asterisk indicating it is required. Below the input, the placeholder text "Name" is visible.
- FAIL ON ERROR:** A toggle switch currently set to "YES".
- NOTIFY ON ERROR:** A toggle switch currently set to "YES".
- ENABLED:** A toggle switch currently set to "NO".
- EVENT FULL CLASS NAME:** A text input field with a red asterisk. Below the input, the placeholder text "Event full class name" is visible.
- ACTION FULL CLASS NAME:** A text input field. Below the input, the placeholder text "Action full class name" is visible.
- HTTP METHOD:** A text input field. Below the input, the placeholder text "HTTP method" is visible.
- API URL:** A text input field. Below the input, the placeholder text "API URL" is visible.
- ACTION ARGUMENTS:** A large text area for defining arguments.
- ACTION CONDITIONS:** A large text area for defining conditions.
- JSON preview:** Two preview areas on the right side of the form, one for "ACTION ARGUMENTS" and one for "ACTION CONDITIONS".

At the bottom of the form, there is a red error bar that says "Invalid fields". To the right of the error bar, there is a checkbox labeled "After save, create another" and a red "Save" button.

The creation form consists of several fields, required for creation of a new event action:

- **Name:**
  - The name of an event actions should be **simple and descriptive**, making it easy to figure out an event action's function from name alone (eg. *Send email to user when new order is created*).

With the name set, the event that triggers the event action and the action that is performed by the event action need to be defined.



The image shows a form titled "Basic information" with the following fields:

- NAME:** Send email when stock total iz below zero
- EVENT FULL CLASS NAME:** App\Containers\Product\Events\ProductsStockUpdatedEvent
- ACTION FULL CLASS NAME:** App\Containers\Mail\EventActions\SendEmailAction

- **Event full class name:**

- A full path (namespace and class name) of the Event class that needs to be triggered to perform the event action (eg. *App\Containers\Product\Events\ProductsStockUpdatedEvent*).

- **Action full class name:**

- A full path (namespace and class name) of the Action class that will run once the Event is triggered. (eg. *App\Containers\Mail\EventActions\SendEmailAction*).

With the name, event and action defined, the three top switches can be toggled on/off.

<div>FAIL ON ERROR</div> <div><input type="checkbox"/> NO</div>	<div>NOTIFY ON ERROR</div> <div><input checked="" type="checkbox"/> YES</div>	<div>ENABLED</div> <div><input checked="" type="checkbox"/> YES</div>
---	---	---

- **Enabled:**

- Toggle an event action as active or inactive. Inactive event actions will never run.

- **Fail in error:**

- When this switch is toggled on, the action will be performed as normal, but will stop **immediately** on encountering an error.

- **Notify on error:**

- When this switch is toggled on, whenever the event action encounters an error, an email is sent to the development team to inform them of a failure when performing the event action.

**Event actions** can also be used to **call external services** via **API** calls when conditions are met.

<div>HTTP METHOD</div> <div>HTTP method</div>	<div>API URL</div> <div>API URL</div>
---	---------------------------------------

- **HTTP Method:**

- The method of the API call. Valid call methods include **POST, GET, PUT, PATCH, and DELETE**.

- **API URL:**


- The URL of the external service API that will be called. Should be in format: **https://[service]/[endpoint]**

**Arguments** and **conditions** (currently) need to be written in **JSON format**.

<div>ACTION ARGUMENTS</div> <div><pre>{   "to": "purchasing@dewesoft.com",   "html": "New consumable order &lt;a href='https://erp.dewesoft.com/orders-consumable/{orderConsumable.id}&gt;&lt;strong&gt;\${orderConsumable.document_number}&lt;/strong&gt;&lt;/a&gt; was created.",   "subject": "\${orderConsumable.document_number} created" }</pre></div>	<div>JSON preview:</div> <div><pre>{   "to": "purchasing@dewesoft.com",   "html": "New consumable order &lt;a href='https://erp.dewesoft.com/orders-consumable/{orderConsumable.document_number}&gt;&lt;strong&gt;\${orderConsumable.document_number}&lt;/strong&gt;&lt;/a&gt; was created.",   "subject": "\${orderConsumable.document_number} created" }</pre></div>
--	--

## • ARGUMENTS:

- Available arguments depend on selected **"action class"**. This data can be provided by people with access to the codebase.
- eg. **SendEmailAction** class requires:
  - **"to"**: the email that will receive the notification email
  - **"html"**: the content of the email in HTML
  - **"subject"**: the subject of the email to be sent

ACTION CONDITIONS   
[{"field": "product.stock\_available", "value": 0, "operator": "<"}]

JSON preview:  

```
[
  {
    "field": "product.stock_available",
    "value": 0,
    "operator": "<"
  }
]
```

## • CONDITIONS:

- Available conditions depend on selected **"event class"**. This data can be provided by people with access to the codebase.
- **The example in the photo checks if a product's stock\_total is lower than 0**
- eg. The **ProductsStockUpdatedEvent** class has access to the product object, so we can build conditions with fields of a product, eg. product's **"stock\_total"**.
  - We can set a condition by making a JSON object that includes product's:
    - **"field"**: Any field of a product model record (sku, name, short\_name, id, ...)
    - **"value"**: The value that the field will be compared to
    - **"operator"**: The operation to run (the way to compare the value and the field)

## AVAILABLE CONDITION OPERATORS:

"=" or "==="	Check if the field and value are the same. (strict comparison by value and by type)
"=="	Check if the field and value are the same.
"!=="	Check if the field and value are not the same. (strict comparison by value and by type)
"!="	Check if the field and value are not the same.
"<"	Check if the field is less than the value.
">"	Check if the field is more than the value.
"<="	Check if the field is less than or equal to the value.
">="	Check if the field is more than or equal to the value.

<b>IN</b>	Check if field is one of values in array. Value must be array here.
<b>NOT IN</b>	Check if field is not one of values in array. Value must be array here.
<b>CONTAINS</b>	Check if field contains the value.
<b>DOES_NOT_CONTAIN</b>	Check if field does no contains the value.
<b>IS_TABLE</b>	Check if the record's table is equal to value.
<b>STARTS_WITH</b>	Check if the field starts with value. Field and value must be strings in this case
<b>ENDS_WITH</b>	Check if the field end with value. Field and value must be strings in this case

# Update existing event action

Existing event actions should currently never be updated by anyone else but the developers. When a need arises for a different functionality, a new event action should be created, deactivating the old one.



# Delete event action

Existing event action should never be deleted, instead being deactivated.