



Processes and automation

ELO Workflow



Table of contents

ELO workflows and forms	3
Basics	3
Ad hoc workflow	5
Introduction	5
Start ad hoc workflow	6
Default workflow	11
Introduction	11
The workflow designer	12
Create workflow templates	17
Node types	28
Edit and manage templates	45
Form-based workflow	56
Introduction	56
The form designer	57
Create form	59
Create templates	61
The toolbar	65
Table attributes	81
Create tab group	84
Cell properties	89
Global form settings	102
Integrate a form into a workflow	107
Save form data	112
Validation	116
Custom styles	124
Structure in ELO	129
Advanced functions	133
Introduction	133
Using scripts	134
Events and global functions	150
End workflows	155

ELO workflows and forms

Basics

ELO includes the following types of workflows:

- Ad hoc workflows
- Default workflows
- Subworkflows

In addition, workflows can be linked to forms.

Ad hoc workflows

Ad hoc workflows are predefined by ELO. In general, ELO offers two types of ad hoc workflows:

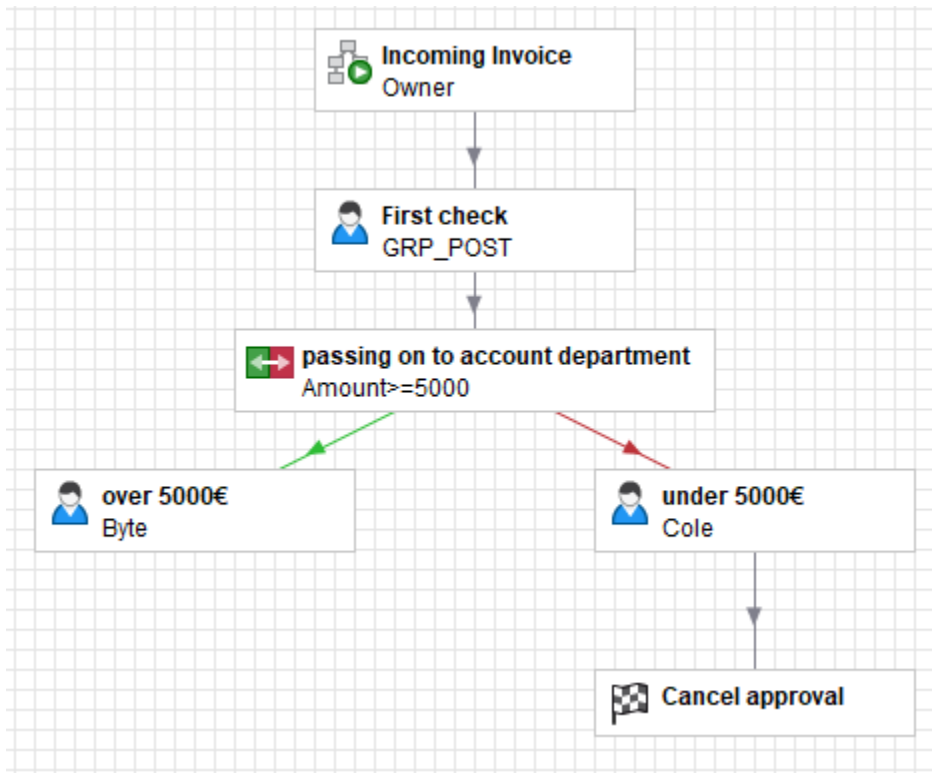
- Approval means that a workflow can only be ended if all responsible participants have marked the assigned task as completed.
- Notification means that the participants of the workflow must confirm the receipt of the workflow. The information associated with the workflow (e.g. a document) is considered to have been received.

You can find more information about the different types of ad hoc workflows in the Ad hoc workflow chapter.

Default workflows

Default workflows cover a large range of company processes. ELO offers workflow templates for this purpose. Configure these templates according to the requirements of the respective process. The workflow templates help to provide standardized templates for all participants.

Invoice approval example



If a company receives an invoice, for example, certain procedures must be followed. In ELO, the responsible employee selects the corresponding template and starts the appropriate workflow. The invoice goes through the necessary steps (nodes) of the workflow and, once it is approved, it is settled.

Subworkflows

Subworkflows have the same structure as default workflows. They are started from within default workflows as soon as the corresponding workflow node has been reached.

Additional information on default workflows and subworkflows can be found in the *Default workflow* chapter.

Forms

In ELO, forms can be used in different locations:

- Form for editing a workflow
- Form as metadata preview
- Form as a substitute for a metadata form
- Form for creating data sets in ELO for Mobile Devices

This documentation focuses on the interaction between workflows and forms.

For more information on creating and editing forms, refer to the Form-based workflow chapter.

Ad hoc workflow

Introduction

Ad hoc workflows are suitable for replicating simple processes. Ad hoc workflows are predefined by ELO and can be set up with relatively little effort. However, these workflows have fewer options than default workflows.

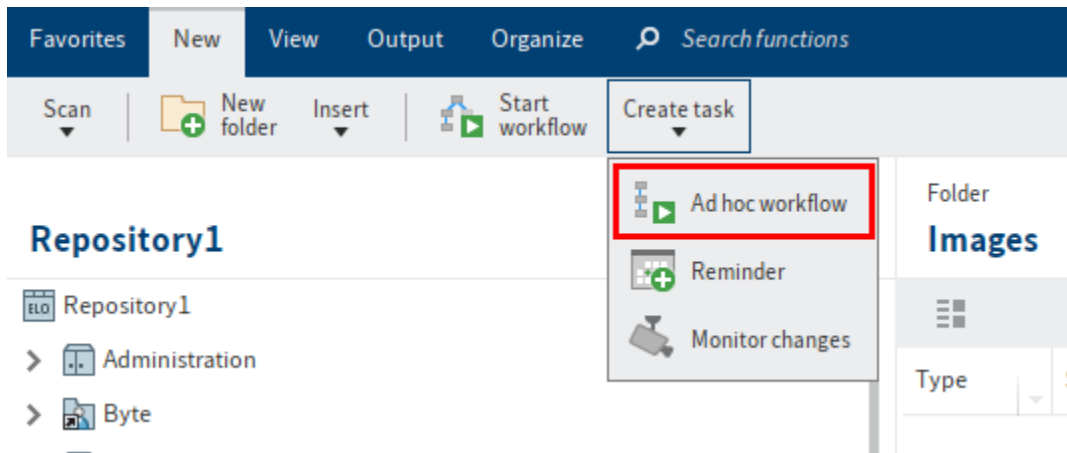
The following ad hoc workflow types are available:

- **Serial approval:** The workflow is passed forward sequentially to the participants. The participants must decide whether or not to approve the workflow step. Depending on the result, the corresponding notification is sent to the selected recipient.
- **Parallel approval:** The workflow is distributed in parallel. All recipients receive the workflow at the same time. Every responsible person must issue approval.
- **Serial notification:** The workflow is passed forward sequentially to the individual participants. The participants must confirm receipt of the workflow one after another. As long as the responsible user does not confirm the workflow, the workflow is not passed forward to the next participants.
- **Parallel notification:** The workflow will be sent to all participants at once. The participants must confirm receipt of the workflow. Once completed, the corresponding notification is sent to the selected recipient.

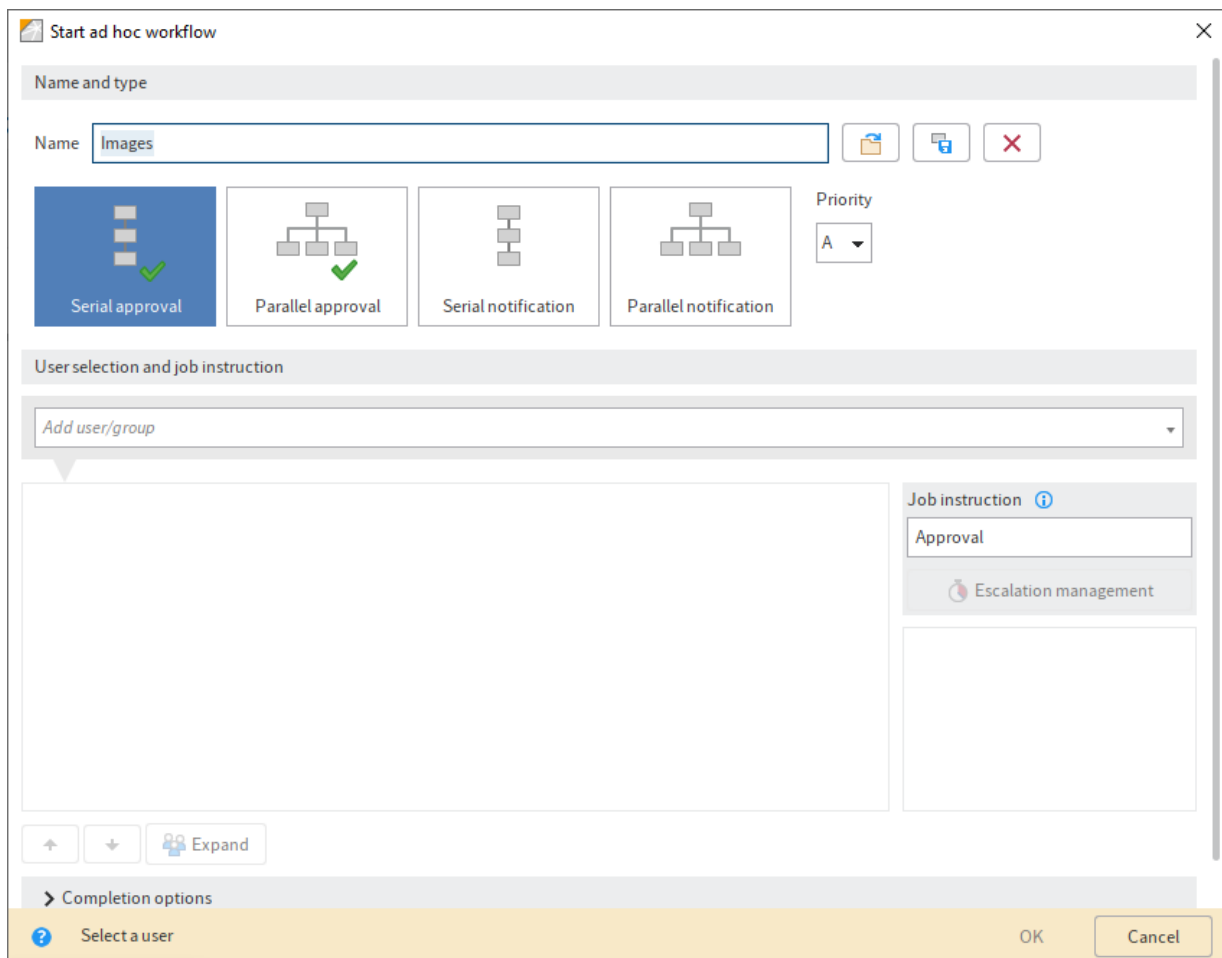
Start ad hoc workflow

Proceed as follows to start an ad hoc workflow:

1. In ELO, select the entry (document or folder) you want to use for an *ad hoc workflow*.



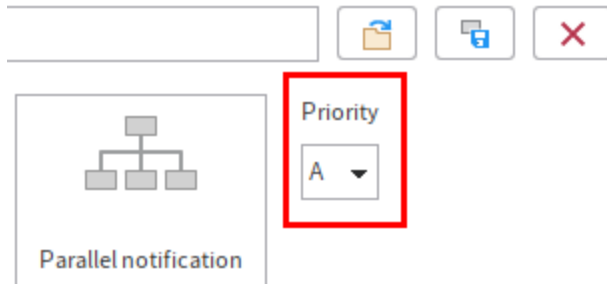
2. Select *Ad hoc workflow* (Ribbon > New > Create task).



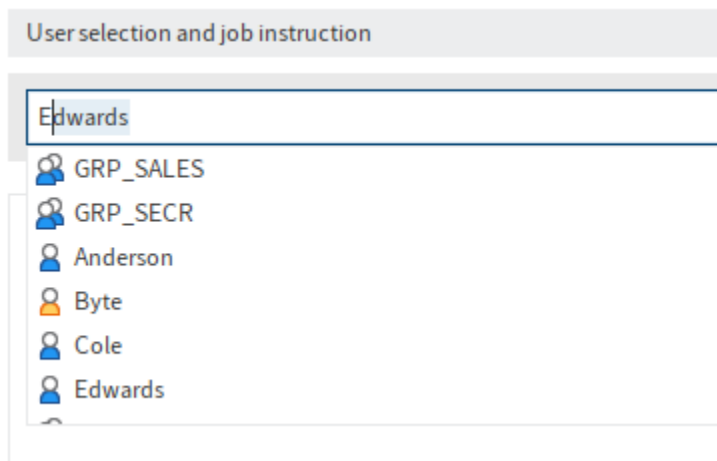
The *Start ad hoc workflow* dialog box opens.

Option 1: You can change the name of the ad hoc workflow in the *Name* field. The workflow is displayed under this name.

- Specify the type of ad hoc workflow.



- You can define the priority level of the ad hoc workflow in the *Priority* drop-down menu. You can choose between the following priorities: A (=High), B (=Medium), and C (=Low). This function is useful if you have a large number of workflows and you want to rate them according to importance.



- Search for the desired user or group in the *Add user/group* field. Suggestions will appear as you type.

Select the relevant suggestion.

Alternative: If you click the triangle to the right of the *Add user/group* field, this will open a drop-down menu. The drop-down menu contains a list of the users and groups you selected recently.

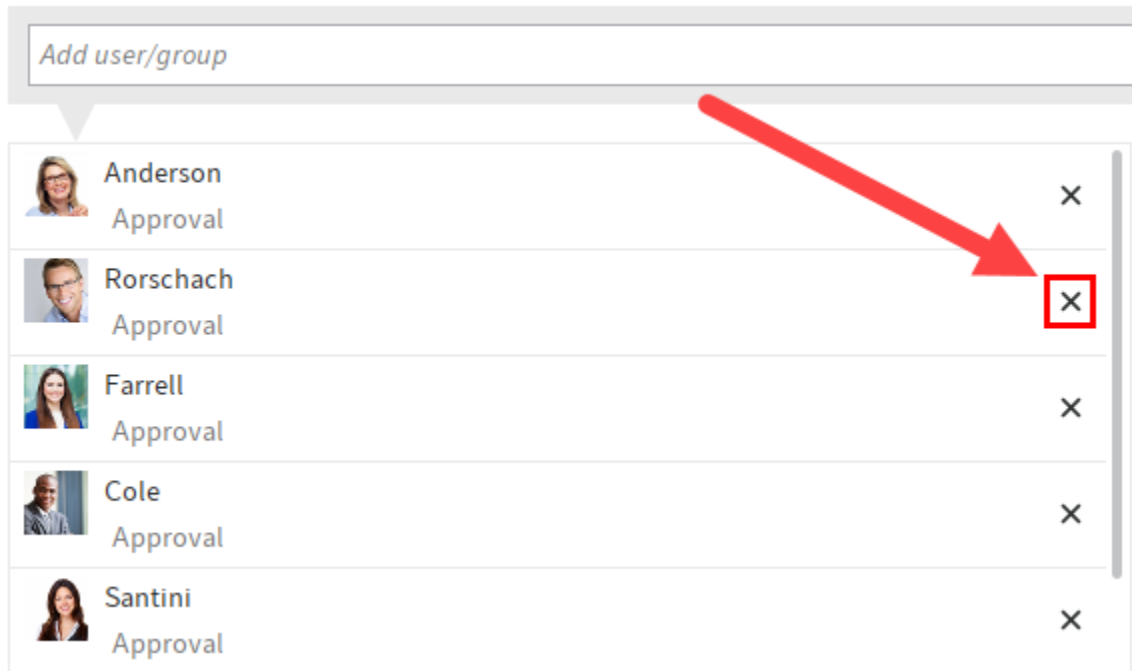
The selection is displayed in the *Add user/group* column. This list is the distribution list for the ad hoc workflow.

Members of the group: If you select a group, you will see the group members in the *Members of the group* area.

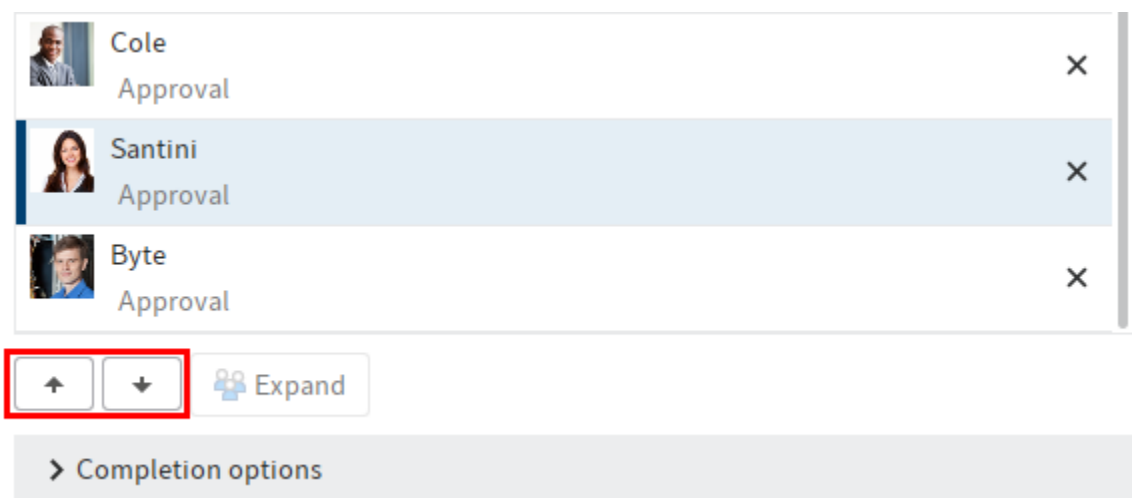
Expand: If you select *Expand*, groups are not displayed as a group. Instead, the individual members of the group are listed.

Information

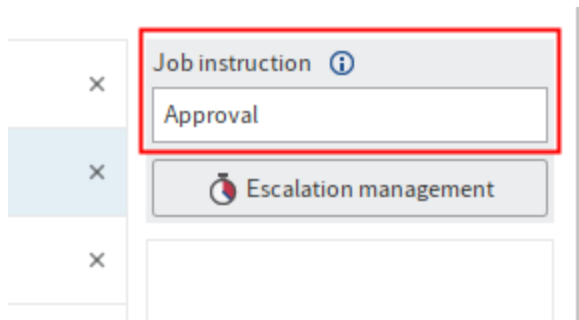
If a group is selected, one member of the group can accept and process the workflow. If you expand the group, every member of the group can accept and process the workflow.



Option 2: Click the X icon next to the corresponding entry to remove the person or group from the list.



Option 3: The order of the participants in the distribution list can be changed. To do so, drag the users to the desired position in the list or use the *Move up* (small up arrow) or *Move down* (small down arrow) buttons.



Option 4: You can change the text in the *Job instruction* field. The respective job instruction applies for the user/group currently selected. You can select multiple users and groups. The job instruction can have a maximum of 128 characters.

Option 5: You can also set the maximum amount of time the ad hoc workflow may remain with the selected user/group. To do so, select *Escalation management*.

Option 6: Select a recipient for the success message when the workflow is completed. To do so, expand the *Completion options* area and then select the *Select user* button.

Option 7: Expand the *Completion options* area and change the message that is shown when the workflow is successfully completed in the *Success message* field. The text also appears as a button when forwarding the workflow.

Option 8: Expand the *Completion options* area and change the message that appears in the *Cancellation message* field if the workflow is canceled. For approval workflows, the text appears as a second button when forwarding the workflow.

Information

You cannot enter a cancellation message for *Notification* workflows.

Option 9: (only applies for *Parallel approval* workflows): Expand the *Completion options* and disable (if required) the option *Withdraw the workflow from all users as soon as one user does not approve it*.

Option 10: Expand the *Completion options* area and select a script to be executed when the workflow is completed from the drop-down menu in the *End script* line.

Information

You can find more information on the topic of *scripts* in the chapter *Advanced functions*.

6. Select *OK*.

The *ad hoc workflow* is started.

Default workflow

Introduction

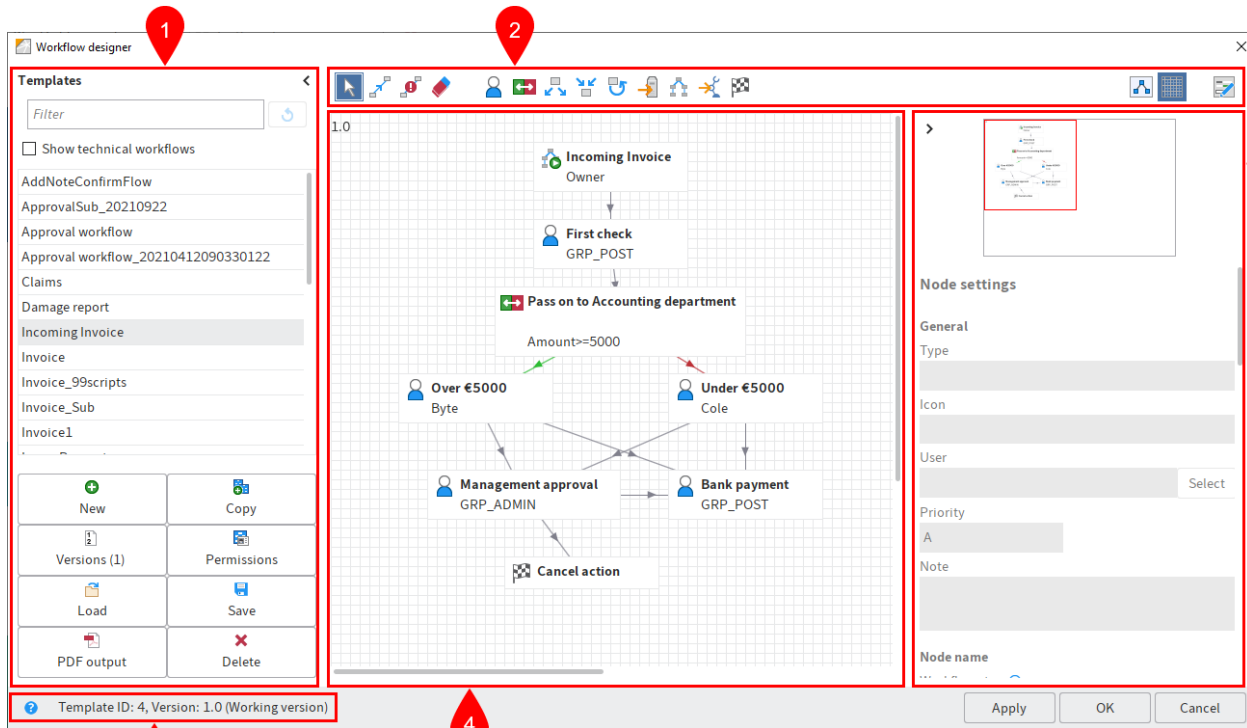
Default workflows are designed for processes that occur often in a company. Default workflows can be used to cover a wide range of requirements. You need a corresponding workflow template to start a default workflow.

You can create and edit workflow templates via the *Workflow designer* dialog box.

This chapter contains the following topics:

- The workflow designer
- Create workflow templates
- Edit node
- Edit and manage templates

The workflow designer



Open the workflow designer via *Ribbon > Organize > System > Workflow designer*.

The workflow designer consists of the following areas:

1 **Templates:** All available workflow templates are listed in the *Templates* column on the left-hand side. Click an entry to open the graphical interface, which displays workflows as a diagram in the middle viewer pane. In addition, the column contains buttons for managing workflow templates:

2 **Toolbar:** This is where you find the tools for creating and editing a workflow template.

Information

When you open a workflow template, the *Edit workflow templates* button appears first. Select *Edit workflow templates* to switch to edit mode.

3 **Node editor:** This area contains a preview window and the *Node settings* area. Define the settings for the individual workflow nodes in the *Node settings* area.

Information

If the start node is selected, the *Workflow settings* area is displayed, where you can adjust the general settings.

4 Workspace: In the workspace, the workflow templates are displayed in the graphic view. You can see the steps (nodes) and elements that a workflow consists of and how they are connected.

5 Status bar: This is where you find the ID of the workflow template, the current version, and the node ID of the currently selected workflow node.

Toolbar



When edit mode is enabled, the buttons for editing workflow templates appear in the toolbar of the workflow designer.

The following functions are available:

1 Select tool

If the *Select* option is enabled, you can drag the nodes of your workflow template to new positions.

If multiple nodes are selected they can be moved together. Select several nodes by selecting one after another while pressing CTRL. It is also possible to draw a frame around the desired node by keeping the left mouse button pressed.

Information

The *Select* function (arrow icon) must be enabled to draw a frame around several nodes.

2 Connection

The *Establish connection between two nodes* function allows you to connect workflow template nodes. The direction of the connection is determined by the order in which you select the nodes. Select the start node first and then the target node.

You can also identify the direction of the connection by the arrow icon in the middle of the connecting line.

Information

You can connect a node to a maximum of 20 successor nodes.

3 Successor node on timeout

The *Specify successor node on timeout* function allows you to connect one node to another, which is automatically called when the deadline is exceeded.

You can set the deadline in the *Node settings* under *Escalation management > General escalation*. If the deadline has passed, the workflow proceeds straight to the respective successor node. Unlike with escalations, the user at the successor node does not have to accept the workflow.

Information

If you use the *Specify successor node on timeout* function, ELO ignores the *Escalation to* field. Instead, ELO passes the workflow on to the successor node.

4 Delete

The *Delete* function removes elements from workflow templates.

If you hover the cursor over an element, an eraser icon appears and you can select the element to delete it.

5 User node

User nodes determine the steps at which the workflow should be processed by a user or a group.

6 Decision node

This node is used to define a condition (if - then) which affects the document flow.

You can use decision nodes to compare fields from a document's metadata form with specific values, for example to check the amount of an invoice.

If the condition is met, the workflow is forwarded to node A (green connection). If the condition is not met, the workflow is forwarded to node B (red connection).

7 Distribution node

You can use a distribution node to send the workflow to multiple successors nodes at the same time.

8 Collection node

A document is only forwarded to the next station by a collection node once all preceding nodes have been completed or only a predefined number of responses is pending.

For example, if an invoice needs to be approved by two employees, it is not forwarded until both employees have approved the invoice.

9 Cycle node

There are processes in workflows that need to be repeated until a specific state is reached. This is what cycle nodes are used for. In the second loop, the node information is not lost because the nodes are copied and inserted next to the existing nodes.

Please note

When using cycle nodes, you need to define a start node (*Cycle start* option) and an end node (*Cycle end* option). Both nodes must have the same name, e.g. CYCLE_1. If several cycles exist within the same workflow template, each cycle must have a unique name.

Please note

The name of nodes must not exceed 128 characters. For cycle nodes, the number of cycles is automatically added to the name of the cycle in the format `[[1]]`. Do not forget these five characters when determining the maximum number of characters.

Only one connection may lead to and from a cycle node, meaning that you may have to use a collection node or a distribution node to connect to multiple nodes.

The end node status determines whether a cycle is passed a second time. However, the entire cycle is already duplicated when it passes the start node, meaning that it is possible to copy nodes within the cycle.

Please note

The nodes within a cycle are not allowed to have connections to nodes outside the cycle.

10 Server transfer

The *Server transfer* node type is used to transfer a workflow document to a second server. The repository ID of the second server must be entered in the server transfer node.

After synchronizing the data to the second server, you can continue processing the workflow on the second server. The workflow is then locked on the first server.

11 Subworkflow

The *Call subworkflow* function allows you to add subworkflow nodes. Once the workflow reaches the subworkflow node, the corresponding subworkflow is started.

The workflow started depends on the template you have selected in the *Select template* drop-down menu.

12 Flow

Add a flow node using the *Flow* function. Flows can be linked with a workflow using a flow node. When the workflow reaches a flow node, one or more flows are called and run, depending on the settings.

Flow nodes can:

-

Trigger events that listen to one or more flows.

- Trigger a specified flow.

13 End node

The end node is used to define a unique end point of a workflow.

You do not have to use an end node. If there is no end node, the workflow ends as soon as there are no more unprocessed nodes. It makes sense to use end nodes if you use a collection node that waits for a specific number of predecessor nodes, for example. When this number is reached, the collection node is passed. However, the workflow remains active since there are still open nodes. An end node fully completes the workflow.

Additional buttons in the toolbar



The following three buttons are also available if the workflow template is not in edit mode. You will find these buttons above the *Node editor* area.

14 Minimized view

This function minimizes or maximizes the graphic display of the workflow. This is helpful when working with complex workflow templates.

15 Use grid

This function shows or hides a grid in the background of the workspace.

16 Open form designer

This function opens the form designer to create forms for using in a form-based workflow.

Information

For more information, refer to the chapter Form-based workflows.

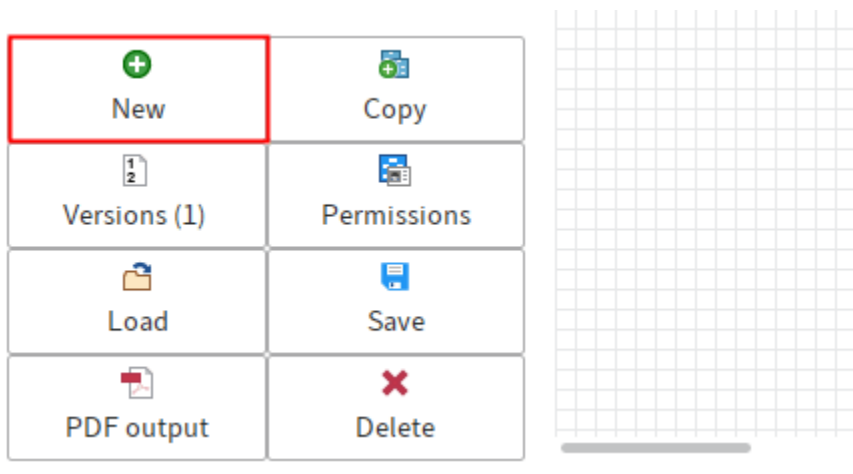
Create workflow templates

Use the workflow designer to create a new workflow template. Processes are mapped using workflow templates. A workflow template must exist if you want to start a default workflow.

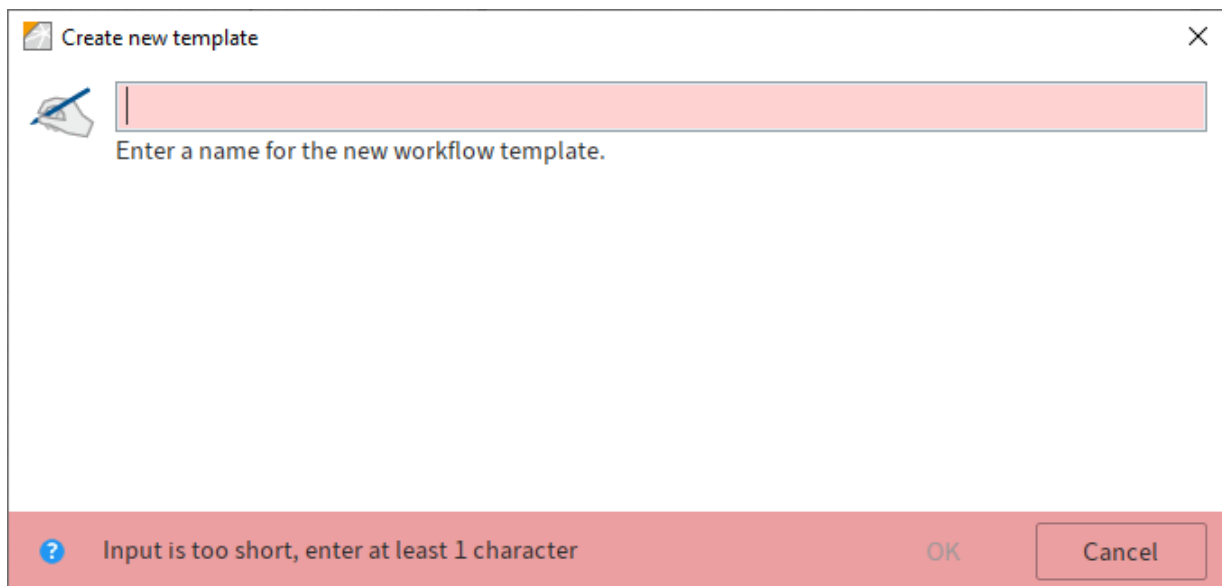
There are two different parts to creating workflow templates. First, create a template. Then configure the template according to your requirements.

Create template

1. Select *Ribbon > Organize > System > Workflow designer*.
2. The *Workflow designer* dialog box opens.



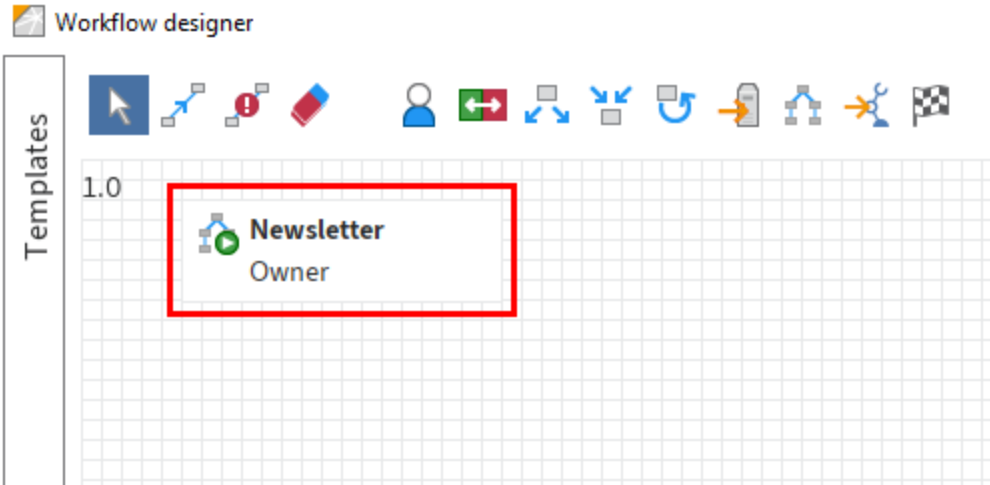
3. Select *New*.



The *Create new template* dialog box opens.

- 4.

Enter a name for the new workflow template. In our example, this is Newsletter. Click *OK* to confirm.



The new template is now available in the *Templates* column. The start node appears in the workspace.

The start node is always set and cannot be deleted. The workflow starts from here.

Please note

The start node may only be connected to one successor node. You cannot link back to the start node.

You have now defined the basic structure of a workflow template. Edit the template according to the requirements of the workflow.

Edit start node

Settings that you make for the start node apply for the entire workflow. In the node editor, the title *Workflow settings* appears instead of the *Node settings* title.

1. Select the start node.

The node is selected.

2. Make the desired settings under *Workflow settings*. The settings are explained in more detail in the following sections.

3. Once you have entered all information, select *Apply*.

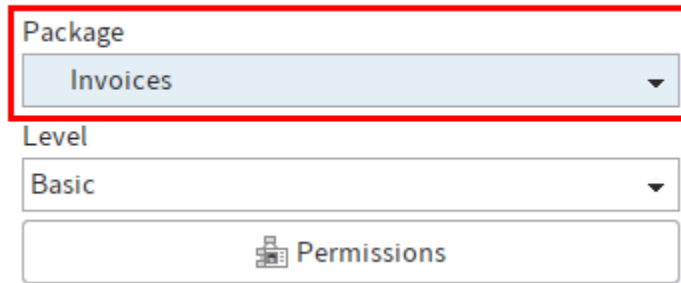
The settings for the start node are saved.

'General' area

In the *General* area, you make settings that apply to the entire workflow template.

Package

General



The screenshot shows a configuration form for a package. The 'Package' dropdown menu is highlighted with a red border and contains the text 'Invoices'. Below it, the 'Level' dropdown menu contains the text 'Basic'. At the bottom of the form is a button labeled 'Permissions' with a document icon.

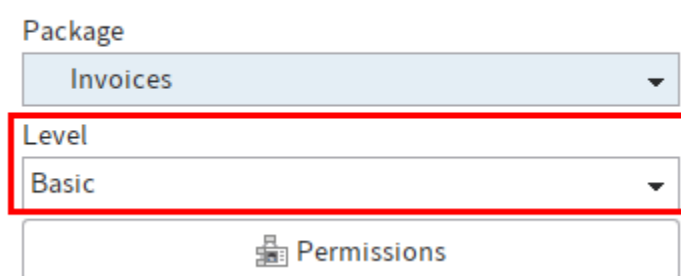
Packages allow you to create and edit related configurations. A package contains all the configurations required for a purpose.

If needed, you can choose which package to assign the workflow to from the *Package* drop-down menu.

For more information, refer to the [ELO packages](#) documentation.

Level

General



The screenshot shows a configuration form for a level. The 'Level' dropdown menu is highlighted with a red border and contains the text 'Basic'. Above it, the 'Package' dropdown menu contains the text 'Invoices'. At the bottom of the form is a button labeled 'Permissions' with a document icon.

Levels are used in packages to realize different customization levels of a basic package. This allows you to manage customizations made in a test system separately from those in a production system, for example.

If you've selected a package, you can choose which level of the package to manage the workflow template at from the *Level* drop-down menu.

For more information, refer to the [ELO packages > Manage levels](#) documentation.

Permissions

Level

Basic

Permissions

Type

Start node

Icon

Select *Permissions* to open the *Workflow permissions* dialog box. In this dialog box, you can edit the permissions settings for workflows that are started with a workflow template.

Icon

Type

Start node

Icon

Start node

Transfer to server ⓘ

Choose a new icon from the *Icon* drop-down menu if needed.

Transfer to server

Icon

Start node

Transfer to server ⓘ

User

Owner

Select

The *Transfer to server* field is where you enter the repository ID of the server you want to transfer the workflow to.

Information

This option is only required when replicating workflows, i.e. if you have installed the ELO Replication module.

User

Transfer to server ⓘ

User

Owner Select

Priority

A ▼

Note

The default setting for the *User* field is *Owner*. *Owner* means: The person who starts the workflow will process all nodes with this setting. You cannot make any changes to the start node here.

Priority

Owner Select

Priority

A ▼

A

B

C

Node name

You can define the priority level of the workflow in the *Priority* drop-down menu. You can choose between the following priorities: A (=High), B (=Medium), and C (=Low). This function is useful if you have a large number of workflows and you want to rate them according to importance.

Note

Priority

A ▼

Note

Node name

Use the *Note* field to enter information on the start node. This text appears in the *Workflow overview*, *Workflows for this entry*, and *Show workflow* dialog boxes.

Node name

Workflow step ⓘ

Newsletter

▼ Translation variable

'Node name' area

In the *Node name* area, you can enter a name for the node. You can also enter a translation variable.

Workflow step

The name of the workflow template is automatically used for the start node. If you want to, you can change the name in the *Workflow step* field.

Translation variable

Use the *Translation variable* field if you want to offer the contents of the *Workflow step* field in multiple languages. Enter the appropriate translation variable.

You will find more information on the topic of translation in the following texts:

- [This documentation > Form-based workflow > Global form settings > Translation variable \(prefix\)](#)
- [Client administration > ELO Java Client > Translation](#)
- [ELO packages preview > Other topics > Translations](#)

'Escalation management' area

In the *Escalation management* area, you can set deadlines for the entire workflow (via the start node) or individual nodes.

Exclude weekends

▼ Escalation management

Exclude weekends ⓘ

General escalation ⓘ

Days	Hours	Min	Escalation to
<input type="text"/>	<input type="text"/>	<input type="text"/>	Owner <input type="button" value="Select"/>

Escalation B

If the option *Exclude weekends* is enabled, the maximum duration of the workflow will take into account that Saturdays and Sundays are not regular business days. Weekends will be skipped when calculating the workflow duration.

If this option is cleared, Saturdays and Sundays are included in the calculation. All calendar days count when calculating the maximum workflow duration.

General escalation

The fields under *General escalation* allow you to define how long the workflow may remain at the current node.

Exclude weekends ⓘ

General escalation ⓘ

Days	Hours	Min	Escalation to
<input type="text"/>	<input type="text" value="6"/>	<input type="text" value="15"/>	<input type="text" value="Owner"/> <input type="button" value="Select"/>

If the workflow has not been completed by the deadline, it shows up in the list of overdue tasks. The user entered in the field *Escalation to* will be sent a message.

Information

With a *General escalation*, the responsible user is selected via the start node and then applies to all nodes.

If you do not enter a maximum duration, there is no automatic check for missed deadlines.

Escalation B

Days	Hours	Min	Escalation to
<input type="text" value="5"/>	<input type="text"/>	<input type="text"/>	<input type="text"/> <input type="button" value="Select"/>

Escalation C

Days	Hours	Min	Escalation to
<input type="text" value="7"/>	<input type="text"/>	<input type="text"/>	<input type="text"/> <input type="button" value="Select"/>

Escalation B

The fields under *Escalation B* allow you to define a second escalation level. If this deadline is passed, the corresponding user is informed.

Escalation C

The fields under *Escalation C* allow you to define a third escalation level. If this deadline is passed, the corresponding user is informed.

'Additional options' area

The *Additional options* areas contains further settings for workflows.

Start workflow manually

▼ Additional options

Workflow can be started manually

Start script ⓘ

Form ⓘ

If the option *Workflow may be started manually* is enabled, the workflow template can be run via the *Start workflow* button. If the option is disabled, the workflow template can only be started as a subworkflow.

Start script

Workflow can be started manually

Start script ⓘ

Form ⓘ

Script properties ⓘ

In the *Start script* field, you can enter a script that will be run when the node becomes active.

Information

Start scripts are run via the ELO Indexserver. You will find additional information on these scripts in the *Advanced functions* chapter.

Form

In the *Form* field, you can link part of a form (template) to the node. The selected templates are displayed when you call the node. Open the *Workflow form selection* dialog box with *Select*.

Script properties

In the *Script properties* text field, you can enter additional properties for scripts.

Add node

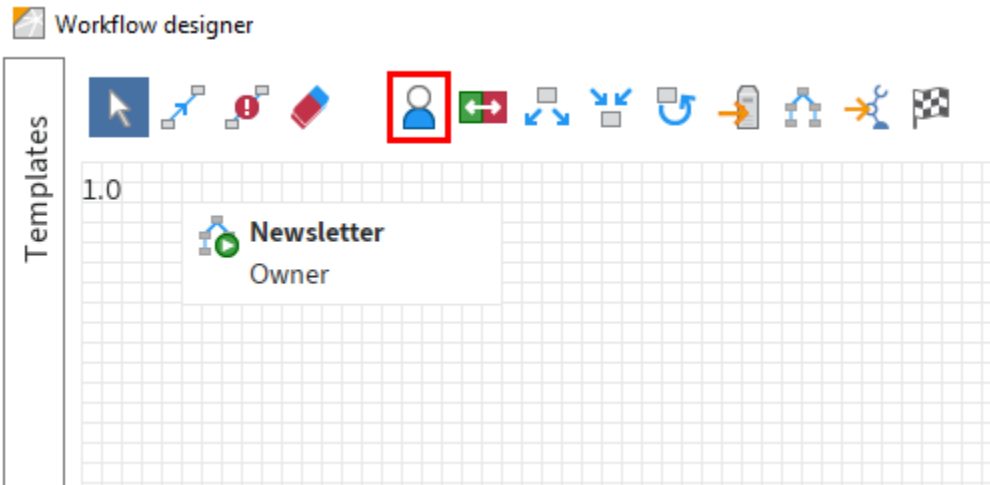
The template requires additional nodes for a workflow to function. The type of node you choose and which settings you make will depend on the type of the workflow.

The node types have different functions and settings.

When you are creating a node, the method is the same for all node types.

In the following, we explain this method based on the user node. You can find more information on the various node properties in the section *Node types*.

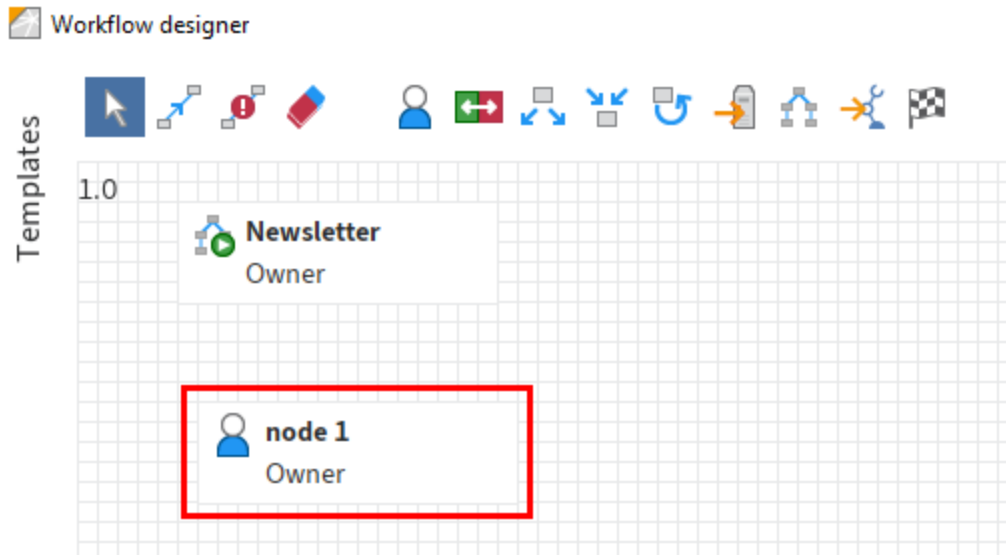
Create user nodes



1. Select the *User node* button.

The cursor turns into the icon for the user node.

2. Place the cursor where you want to insert the user node.
3. Click the left mouse button.



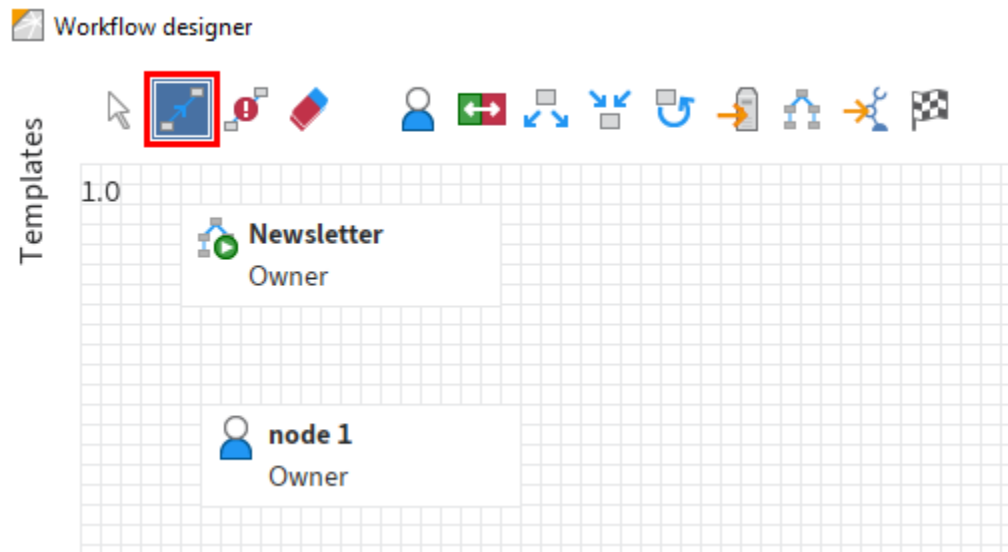
The user node is inserted. The name of this node is *Owner* by default.

Link nodes

Use the *Establish connection between two nodes* function to connect nodes within a workflow.

Please note

The start node may only be connected to one successor node. You cannot link back to the start node.



1. Select *Establish connection between two nodes*.

The cursor turns into a node connector icon.

2. Select the start node, i.e. the node that the connection should start from.

The node is selected.

3. Select the target node.

The workflow nodes are now connected. The arrow icon indicates the direction of the connection.

Information

It is possible to link nodes in both directions. In this case, the connecting line has two arrows.

Complete template

Once you have created all nodes and made all settings, you can complete the workflow template with *OK*.

Alternatively: To save the template without closing the workflow designer, select *Apply*.

The new workflow template is saved and can be used to create a default workflow.

Information

The *Save* and *Version* functions provide options for saving the template. Refer to the *Edit and manage templates* section for more information on the two functions.

Node types

You can make different node settings depending on the node type.

Open node

Open the respective node as follows to edit the settings of a node:

1. Select the node that you want to edit.

Make the desired settings in the *Node settings* area. Some fields correspond to the fields of the start nodes.

The following section describes the most important settings for each node type:

User node

The following fields have been added or differ from the start node.

General

Type
Usernode

Icon
Usernode

User
Owner

Priority
A

User: In the *User* field, select who should process the node. The default for this field is *Owner*.
Owner means: The person who starts the workflow will process all nodes with this setting.

Use *Select* to choose another user or a group.

Icon
Usernode

User
GRP_SECR

Select second group
GRP_POST

Link group

Priority

Select second group: Add another group if required. To do so, use the *Select* button to the right of the *Select second group* field. The two groups will be connected as an AND group. Only users who are members of both groups will receive the workflow.

Select second group

Link group

Priority


Note

Node name

Link group: If you have selected the same group as the user for multiple nodes, these nodes can be linked. All linked group nodes are assigned to the same group member who has accepted the workflow. In this way, you can avoid different members of a group having to take action on the same workflow.

Link group nodes:

1. To link multiple group nodes, select *Link group nodes*.

 Link group nodes
×

Link multiple nodes of the same group. If a user accepts a node of the group, that user is automatically assigned the other group nodes. A node can only belong to one group.

Links

Concept ID: 1 [will be added]

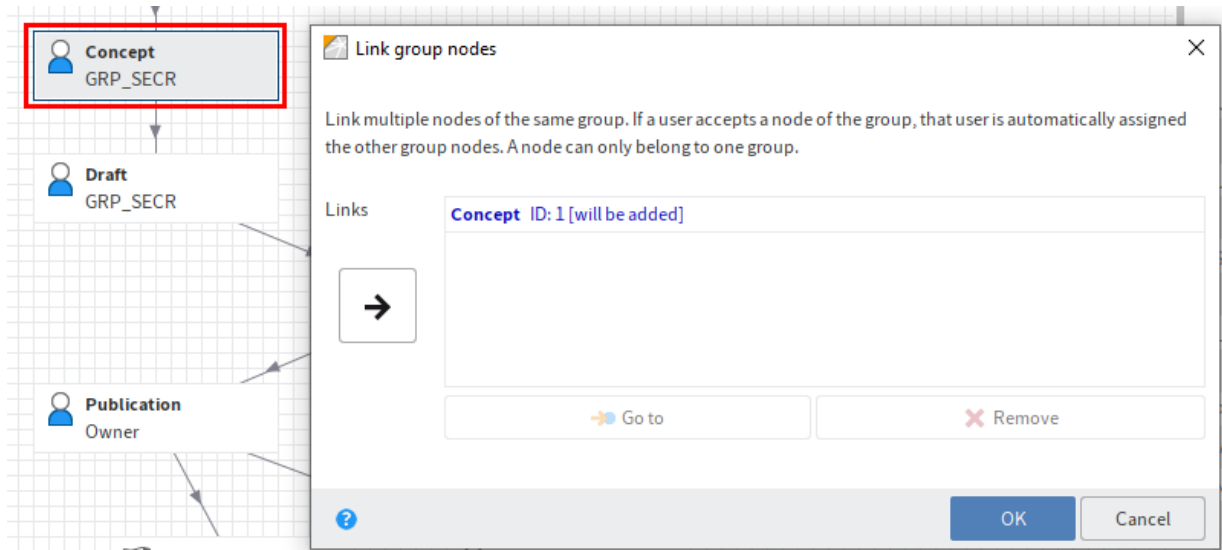
→

→ Go to

✕ Remove

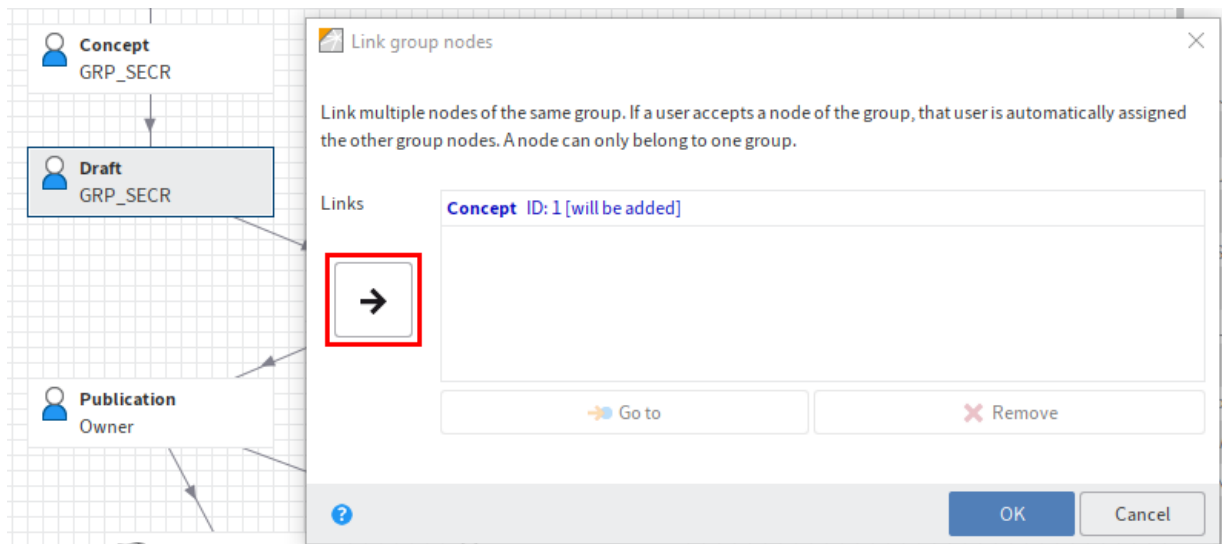
?

The *Link group nodes* dialog box opens.



2. In the workspace of the workflow designer, select the node you want to link to the node you have previously selected.

The *Link group nodes* dialog box remains open.



3. In the *Link group nodes* dialog box, select *Link selected group nodes* (right arrow).

The selected node is added to the list. Each node within the link has a unique ID.

4. Click *OK* to close the dialog box.

In the *Link group* field, you will see the number of the link group that the respective node belongs to.

Link group nodes

Priority
A

Note

Node name
Workflow step ⓘ

Note: You can enter a message for the responsible editor of the workflow node, such as a job instruction or a specific note, in the *Note* text field.

Node name
Workflow step ⓘ

Draft

Translation variable

Name on forwarding ⓘ

Complete concept

Translation variable

Name on forwarding: In the *Name on forwarding* field, you can enter a different name to the one in the *Workflow step* field. The contents of the *Name on forwarding* field are applied to the button label in the *Forward workflow* dialog box.

If you leave the *Name on forwarding* field blank, ELO automatically applies the contents of the *Workflow step* field to the label for the button on forwarding.



Information

The *Name on forwarding* field is available for all nodes with the exception of the start node.

Translation variable: The *Translation variable* field is required if you need the contents of the *Name on forwarding* field in multiple languages. Enter the corresponding key from the respective properties file.

For more information, refer to the chapter *Form-based workflow > Global form settings under Languages* or *Translation variable (prefix)* and the chapter [Translation](#) in the documentation *ELO Java Client administration*.

▼ Forwarding sequence

 Bank payment	↑
 Cancel action	↓

Forwarding sequence: If a user node has several successor nodes, you can specify the order of appearance of the successor nodes when forwarding the workflow. This affects the buttons in the *Forward workflow* dialog box.

Use the *Forwarding sequence* field for this. Use the arrow icons to change the position of the successor node selected in the field.

▼ Additional options

Start script ⓘ

End script ⓘ

Show after number of days/Postponed

Only one successor node possible ⓘ

Reset successor nodes ⓘ

End script: In the *End script* field, you can define an action that is carried out when the workflow is forwarded. For example, you can send a status message to the workflow owner.

Information

End scripts are run via the ELO Indexserver. You will find additional information on these scripts in the *Advanced functions* chapter.

Show after number of days/Postponed

Only one successor node possible [i](#)

Reset successor nodes [i](#)

Form [i](#)

Action buttons [i](#)

Show after number of days/Postponed: Use the *Show after number of days/Postponed* field to delay the workflow. If the workflow is postponed, it only shows up in the responsible user's tasks list once the specified number of days have passed. The postponement is triggered as soon as the workflow has been forwarded to the node.

Please note

The number of days for the postponement should not exceed the maximum node duration. Otherwise, the workflow will pass the deadline before it can be processed.

Show after number of days/Postponed

Only one successor node possible [i](#)

Reset successor nodes [i](#)

Form [i](#)

Action buttons [i](#)

Only one successor node possible: If the option *Only one successor node possible* is enabled, only one node can be selected when forwarding.

End script [i](#)

Show after number of days/Postponed

Only one successor node possible [i](#)

Reset successor nodes [i](#)

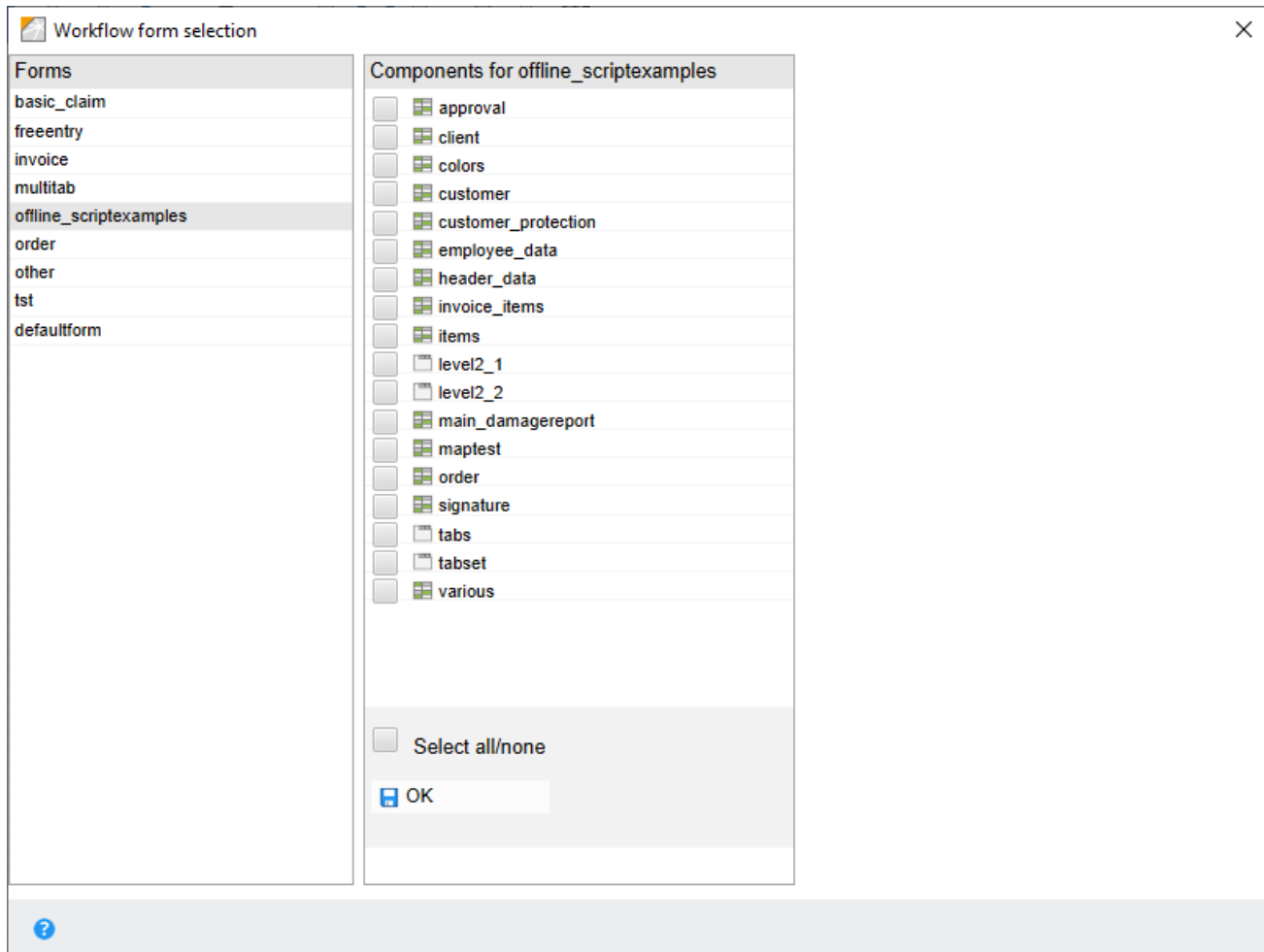
Form [i](#)

Reset successor nodes: If the *Reset successor nodes* option is set, the completed status of all the successor nodes of a workflow node is removed if the branch of a workflow repeats within a cycle. The successor nodes are restored to the original status they had in the first cycle.

Information

Because it is now possible to map a cycle with the help of cycle nodes, this option should only be used in exceptions.

Form: In the *Form* field, you can integrate parts of a form (template) into a node.



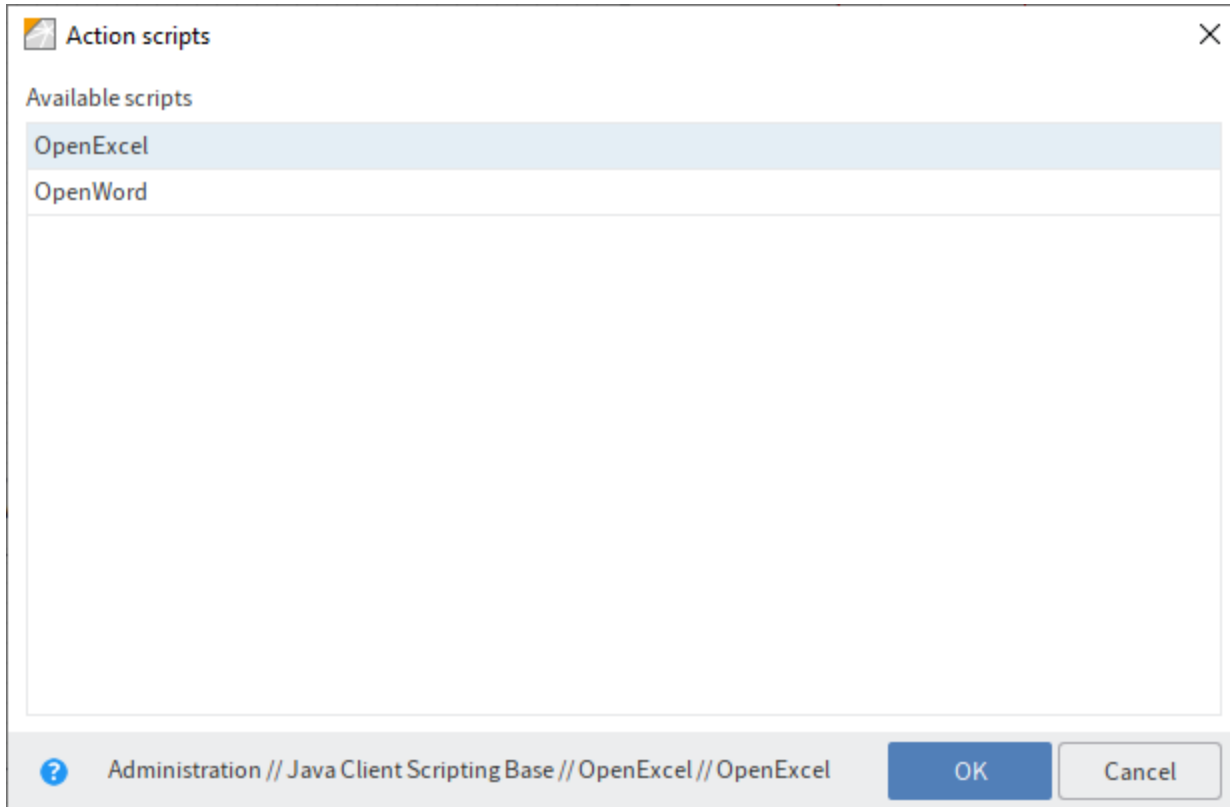
Use *Select* to open the *Workflow form selection* dialog box. This is where you can select the desired templates.

Action buttons: In the *Forward workflow* dialog box, up to five action buttons can be displayed.

Information

If you want to use action buttons, you have to enter at least two action scripts.

The action buttons trigger certain processes, such as sending an e-mail or opening a document.



To open the *Action scripts* dialog box, use the *Select* button next to the *Action buttons* field. Scripts can be selected in the *Action scripts* dialog box.

Forward workflow

Finish editing the current node

Action buttons: New Excel document, New Word document

Information on the current node: Workflow step: Concept, Comments ⓘ

Forward

Select the next workflow step:

→ Complete concept

Cancel

The integrated action buttons are displayed in the *Forward workflow* dialog box.

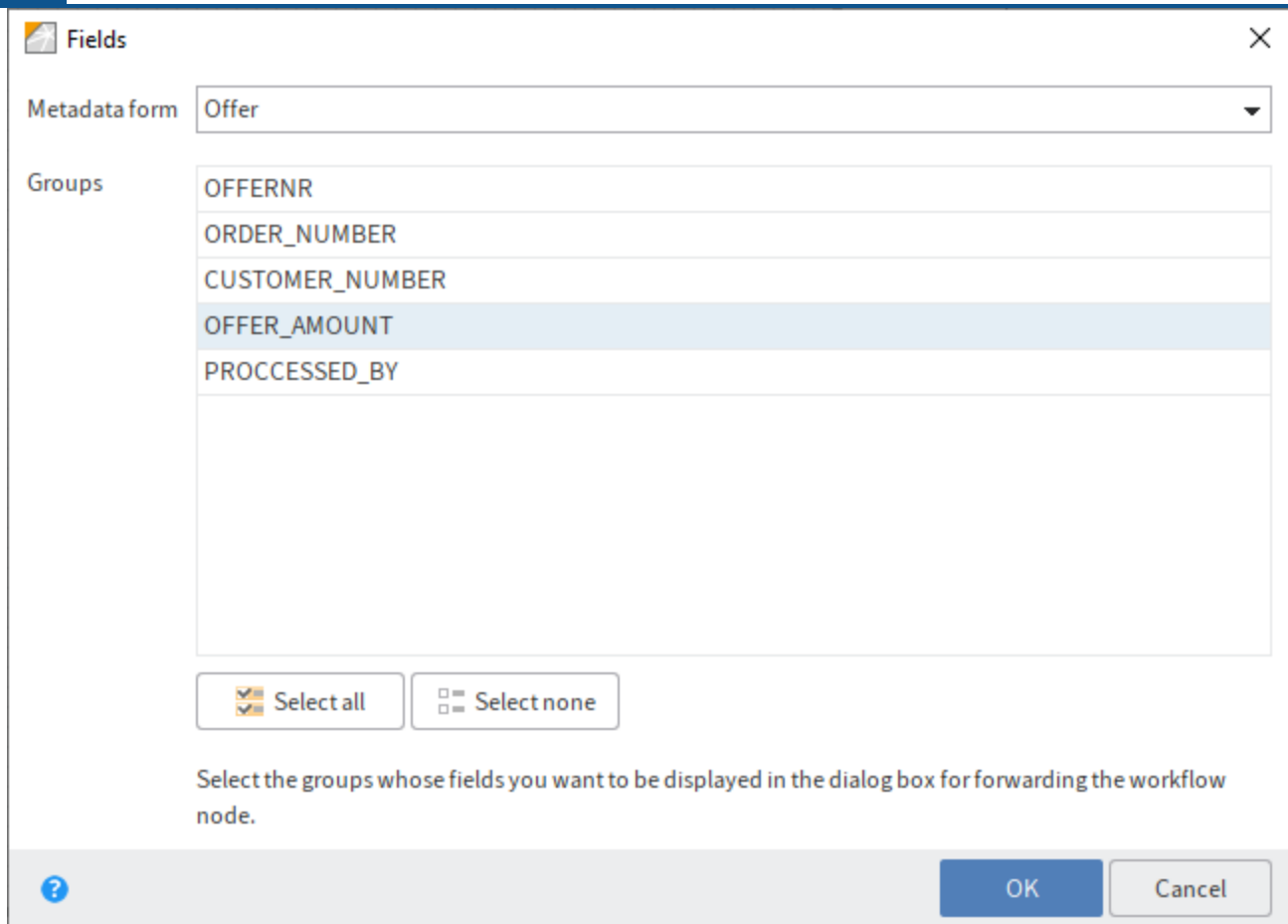
Information

Certain rules apply for action scripts. You will find more information on this topic in the chapter *Advanced functions*.

Field: Use *Field* to select which fields of a metadata form to use for this node. The user who processes the respective node sees the selected fields when forwarding the workflow. This allows you to provide the user processing the workflow with important information. At the same time, the responsible user is also given the option to modify the metadata.

Information

You can select multiple filters. To do so, either press and hold the CTRL key or the SHIFT key, or select the *Select all* option.



Fields

Metadata form: Offer

Groups:

- OFFERNR
- ORDER_NUMBER
- CUSTOMER_NUMBER
- OFFER_AMOUNT
- PROCESSED_BY

Select all Select none

Select the groups whose fields you want to be displayed in the dialog box for forwarding the workflow node.

OK Cancel

Click *Select* next to *Field* to open the *Fields* dialog box. You can select the desired metadata form from a drop-down menu.

The *Groups* field shows you the field groups used on the metadata form. Use these groups to create links to the respective field in the metadata.

Information

You can see which field in the metadata is linked to which group in the ELO Administration Console.

Decision node

A decision node is used to check the data entered into a metadata form. This data specifies what happens next in the workflow.

The following fields only exist for decision nodes:

Condition: Use the *Condition* drop-down menu to select a comparison operator. The following comparison operators are available:

- equal to (=)
-

- not equal to (<>)
- greater than (>)
- less than (<)
- greater than or equal to (>=)
- less than or equal to (<=)

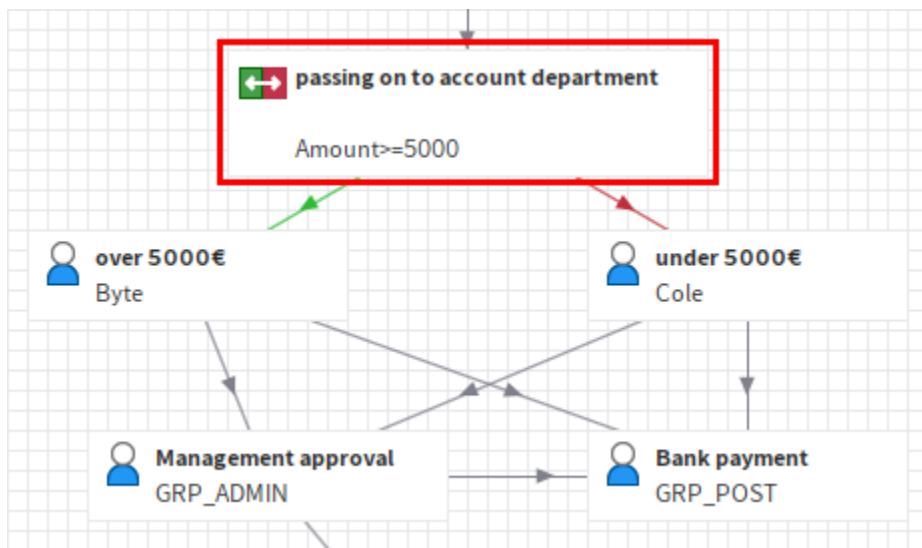
Information

The node must be connected to a field in the metadata for the comparison to work.

Value: Enter the comparison value in the *Value* field. This is compared with the value in the selected field in the metadata using the comparison operator in the *Condition* field.

Depending on the result, the document is forwarded to one or the other successor node.

This means that a decision node must be connected to exactly two successor nodes.

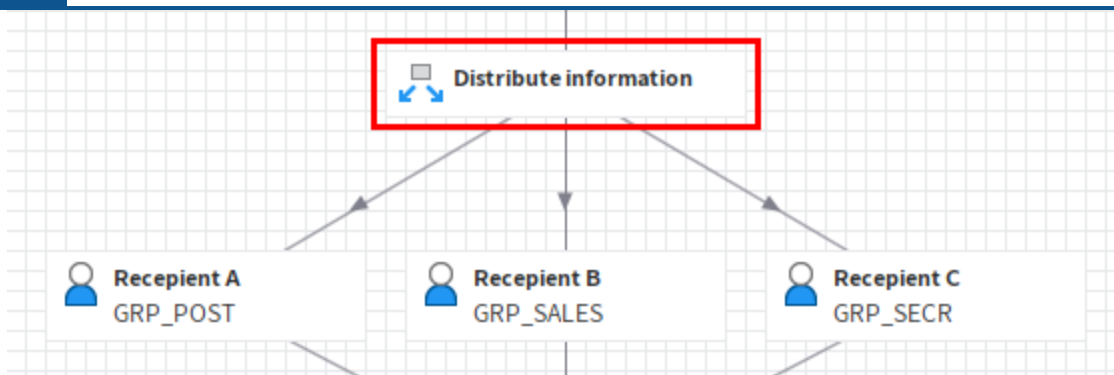


The first result determines the path that is followed if the comparison is TRUE. This connection line is green.

The second connection line is red. This path is followed if the result of the comparison is FALSE.

Distribution node

If you want to forward the workflow to multiple nodes, you need to insert a distribution node.



Status: If you enter a value in the *Status* field, the workflow is assigned the corresponding status as soon as it passes through the distribution node.

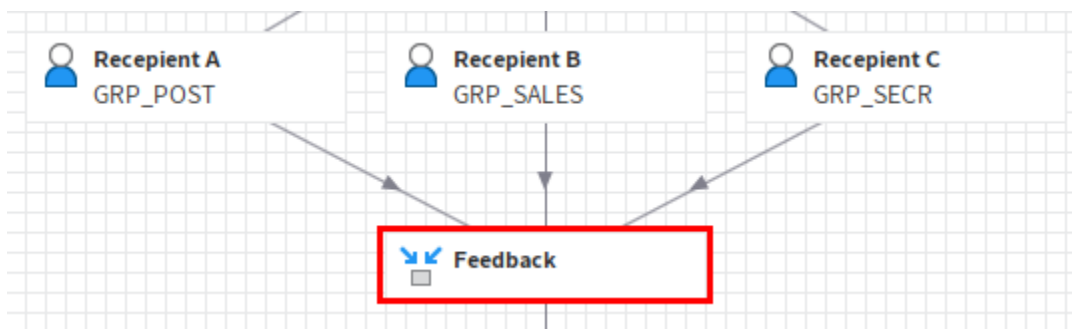
Information

The workflow status can also be modified in ELOas or using scripts.

To query the workflow status, ELO offers the predefined query `ELO_WF_STATUS`. This represents a fixed property that is available for all default workflows. See also *Cycle node*.

Collection node

Collection nodes wait until all preceding nodes have been completed or only a set number of responses is outstanding. The document will not be forwarded until this occurs.



The following options are available for the collection node:

Wait for all predecessor nodes: If the *Wait for all predecessor nodes* option is enabled, the workflow is only forwarded to the next node when all nodes preceding the collection node have been completed.

Forward at the number of completed predecessor nodes: In the *Forward at the number of completed predecessor nodes* field, enter how many predecessor nodes need to be processed before the workflow is forwarded. This means not all predecessors must have processed the node.

Deactivate these nodes when forwarding: Use the field *Deactivate these nodes when forwarding* to determine which nodes should no longer be active once the workflow is forwarded. Enter the respective node IDs.

Information

You see the ID of the selected node in the status bar. Use the *PDF output* button to get a PDF overview with all nodes and corresponding IDs.

Cycle node

Cycle nodes are used if a process has to be repeated until a certain status has been reached.

For each cycle, you need to create a separate cycle node with the option *Cycle start* and a cycle node with the option *Cycle end*. The name must be identical for both of these nodes.

All nodes between *Cycle start* and *Cycle end* will be passed until the desired status has been reached.

When the workflow reaches a *Cycle start*, the affected nodes are duplicated and inserted again with a space. Use the *Spacing* node setting (only in the start node) to define the spacing of the duplicated cycle in the workflow diagram. The higher the number entered, the greater the space between the duplicated cycles.

Similar to the decision node, you define a condition and a comparison value in the *Cycle end*. If the condition is not met, the cycle has to be repeated. If the condition is met, the cycle is ended. For the comparison to work, you need to connect a field in the metadata to the node which the corresponding value is read from.

Alternative: You can also enter the query *ELO_WF_STATUS* in the *Field* field. Use this query to read the workflow status and to check it against the comparison value entered in the cycle node.

Information

You can change the workflow status using distribution nodes, ELOas, or scripts.

Server transfer

The *Server transfer* node type is used to transfer a workflow document to a second server. The repository ID of the second server must be entered in the server transfer node.

After synchronizing the data to the second server, you can continue processing the workflow on the second server. The workflow is then locked on the first server.

Information

This option is only required when replicating workflows, i.e. if you have installed the ELO Replication module.

Subworkflow node

Once the workflow reaches the subworkflow node, the corresponding subworkflow is started.

Type ⓘ

Call subworkflow

Icon

Call subworkflow

Select template

ApprovalSub

Note

Select template: The workflow started depends on the template you have selected in the *Select template* drop-down menu.

All workflow templates can be used for subworkflows. It is also possible to set workflow templates so that they can only be started as subworkflows. To do so, disable the option *Workflow can be started manually* in the start node of the respective subworkflow.

Workflow overview

Status

Active Passed deadlines only

Completed Load fields

All workflows

User

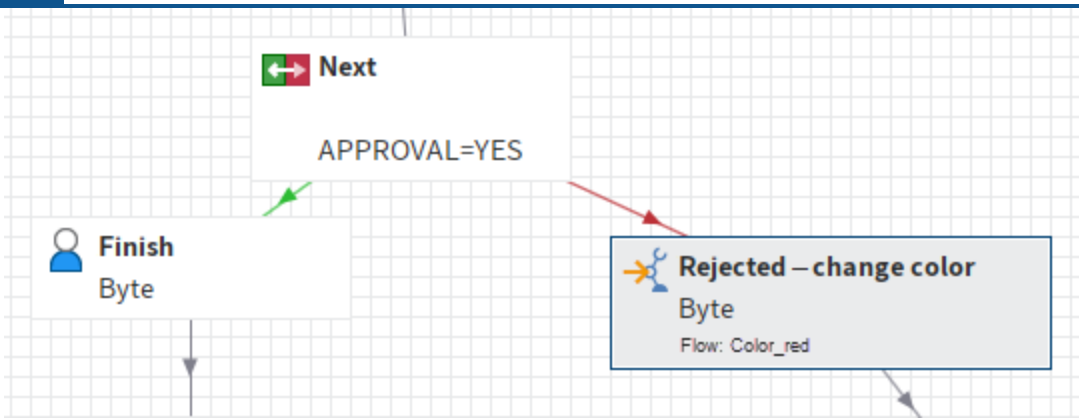
All users and groups

Name ▲	Template	Type	Start date	Duration
Newsletter summer 2021	Newsletter	Main	Yesterday 2:55 PM	20 Hours 4 Min
Newsletter summer 2021_2...	ApprovalSub_20210..	Sub	Today 10:55 AM	4 Min

In the *Workflow overview* dialog box, you can distinguish between default workflows and subworkflows in the *Type* column. Default workflows have the type *Main*. Subworkflows have the type *Sub*.

Flow node

Flows can be linked with a workflow using a flow node. When the workflow reaches a flow node, one or more flows are called and run, depending on the settings.



Event: If the *Event* option is enabled, all flows that listen to this event are called when the flow node is reached.

You can specify a name for the event in the *Template* field (see below).

Flow: If the *Flow* option is enabled, a specific flow is called.

Select the template for the flow via the *Template* field (see below).

Template: In the *Template* field, you specify which flow/event to trigger once the workflow reaches the flow node.

The following variants are possible:

- If the *Event* option is enabled (see above), enter the name of the event you want to trigger. To be triggered, the event has to be configured in one or more flows templates as an event in the trigger.
- If the *Flow* option is enabled, select the desired flow template from the drop-down menu.

Error user: Enter an ELO user in the *Error user* field. The workflow is forwarded to this user if an error occurs executing the flow.

Information

The *Error user* field is only active if the *Flow* option has been enabled.

End node

Use an end node to end a workflow at a fixed point. When the workflow reaches an end node, the workflow is automatically ended. Unprocessed nodes are ignored.

Information

If there is no end node, the workflow ends as soon as there are no more unprocessed nodes.

Return value: The *Return value* field is intended for subworkflows. Enter a successor node to be selected when the subworkflow is completed.

If you work with translation variables, enter the translation variable of the successor node. If you don't use translation variables, enter the node name.

Node settings overview

The following table shows which settings can be made for which node.

Setting	Node
Select second group	User node
Action button	User node
Workflow step	All nodes
Wait for all predecessor nodes	Collection node
Condition/Value	Decision node, cycle end
User/Group	User node
Permissions	Start node
Name on forwarding	All except start node
Deactivate node	Collection node
Level	Start node
End script	All except start node and flow node
General escalation	Start node, user node, subworkflow node
Escalation B and C	Start node, user node, subworkflow node
Event	Flow node
Error user	Flow node with <i>Flow</i> option enabled.
Fields	User node, decision node, cycle end
Flow	Flow node
Reset successor nodes	User node
Form	Start node, user node
Note	All nodes
Only one successor node	User node
Package	Start node
Priority	Start node
Forwarding sequence	User node
Return value	End node
Exclude weekends	Start node, user node, subworkflow node
Show after...	User node
Script properties	All nodes except flow node
Start script	Start node, user node, cycle node, subworkflow node

Setting	Node
Icon	All nodes
Transfer to server	Node, server transfer node
Translation variable	All nodes
Link group	User node
Spacing	Cycle start
Template	Flow node
Select template	Subworkflow node
Forward at the number...	Collection node
Workflow can be started manually	Start node
Duplicate cycle at start	Cycle start

Edit and manage templates

Every workflow template can be subsequently edited. For example, you can add, move, or delete nodes. Furthermore, the workflow designer offers you the option to add and manage versions of workflow templates.

Information

If the *Start workflow in edit mode* option is selected, you can still edit workflow templates when the workflow has started. You can find the option under *Ribbon > <Name of the ELO account used> > Configuration > Advanced settings > Workflow*.

You can edit workflow templates via the *Workflow designer* dialog box. For most editing actions, you need to first enable edit mode.

Edit mode

1. Open the workflow designer via *Ribbon > Organize > System > Workflow designer*.

The *Workflow designer* dialog box opens. All existing workflows are listed under *Templates*.

2. Select the workflow template you want to edit.

The *Edit workflow template* button is enabled.

3. Select *Edit workflow templates*.

The toolbar for editing the workflow template appears.

Move nodes

Nodes can be moved once you have selected a template and enabled edit mode. Use the default mouse pointer (white arrow icon) to do so.

Optional: If required, activate the default mouse pointer via the *Select* button (default mouse pointer) on the toolbar.

1. Select the node that you want to move and drag it by holding down the left mouse button.

Information

Existing connections are retained.

Delete nodes and connections

Nodes and connections can be deleted once you have selected a template and enabled edit mode.

1. Select *Delete* (eraser icon) on the toolbar.

The cursor turns into an eraser icon.

2. Select the element (node or connection) you want to delete.

The selected element is deleted without showing a confirmation request.

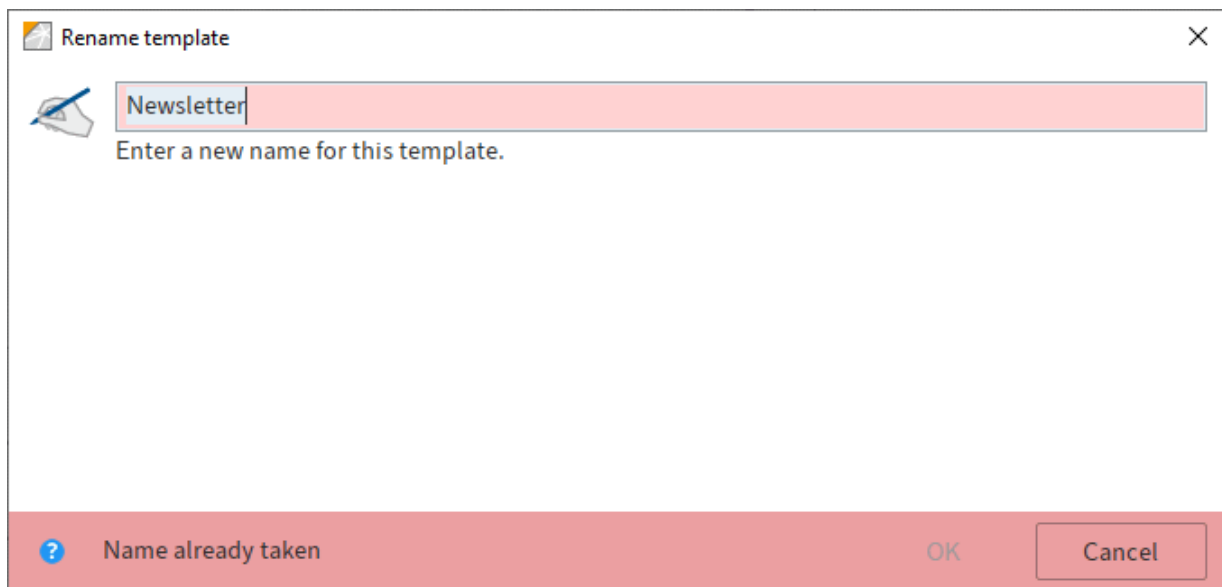
Information

The start node cannot be deleted.

Rename workflow

A workflow template can be renamed as follows if you have selected a template and enabled edit mode:

1. Double-click the name of the respective template in the *Templates* column.



The *Rename template* dialog box opens.

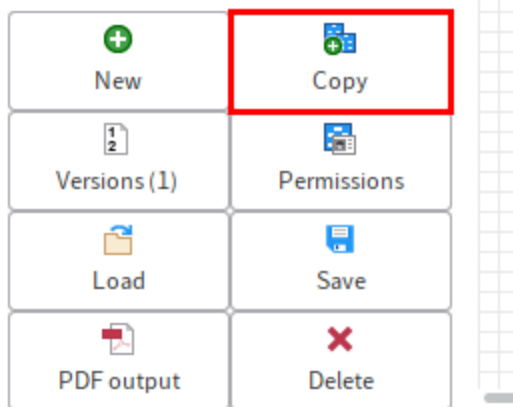
2. Change the name of the template.
3. Select *OK*.

The new name of the template appears in the *Templates* column.

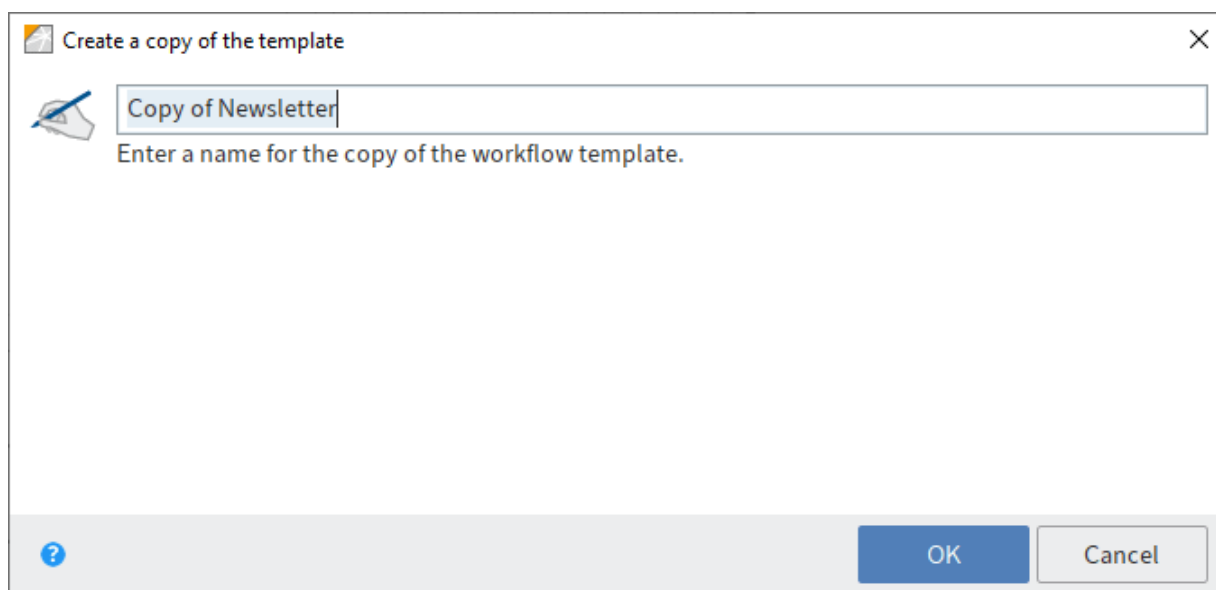
Copy template

Workflow templates can be copied as follows:

1. Select the template that you want to copy in the *Templates* column.



2. Select *Copy*.



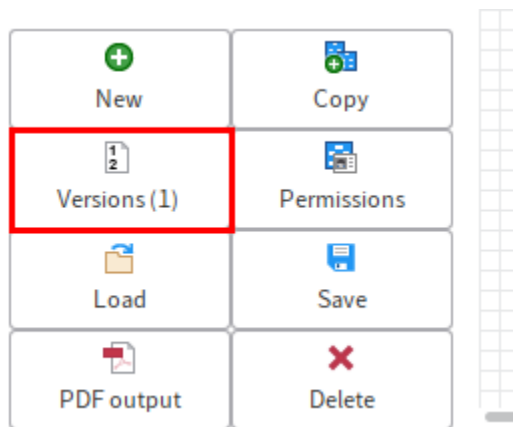
The *Create a copy of the template* dialog box opens.

3. Enter a name for the copy of the template.

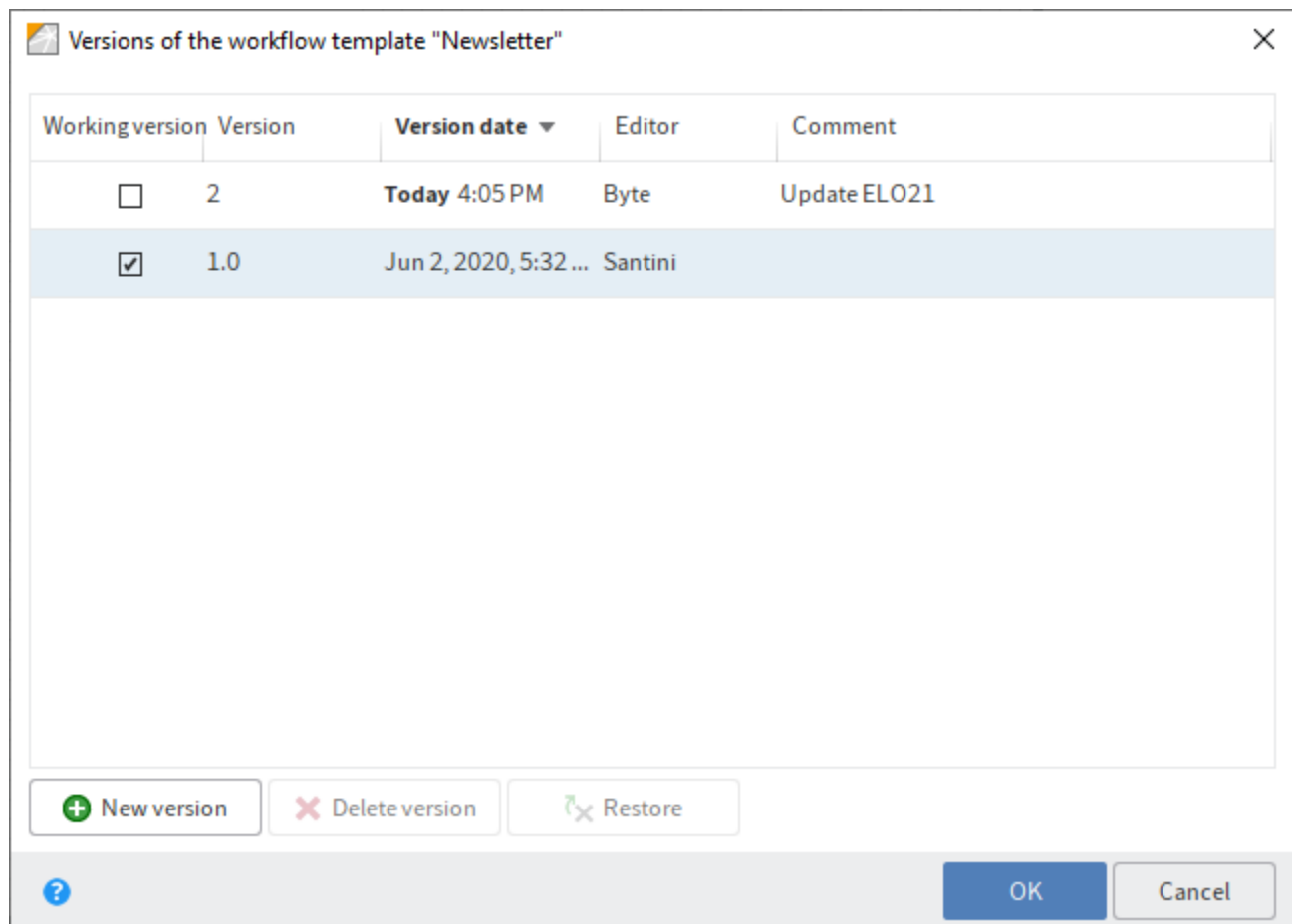
4. Select *OK*.

The copied template appears in the *Templates* column.

Versions



Once you have selected a workflow template and enabled edit mode, you open the *Versions of the workflow template* dialog box via *Versions*.



In this dialog box, you have the following options:

Working version: The working version is the version that has the check box enabled in the *Working version* column. If you select the check box of another version, this version becomes the working version.

Version date: You see when which version was created in the *Version date* column.

Editor: You see who created the respective version in the *Editor* column.

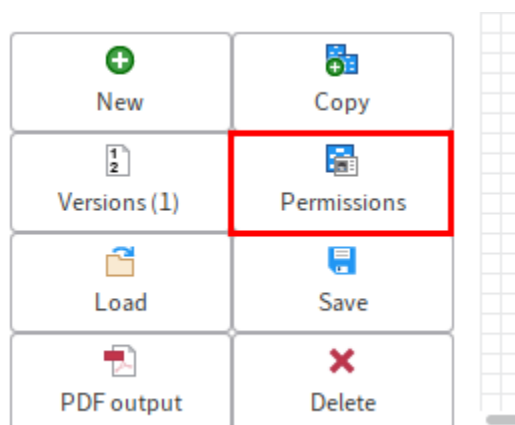
Comment: Any version comments are listed in the *Comment* column. Double-click a comment to edit it.

New version: *New version* allows you to save the current state of the workflow template as a new version.

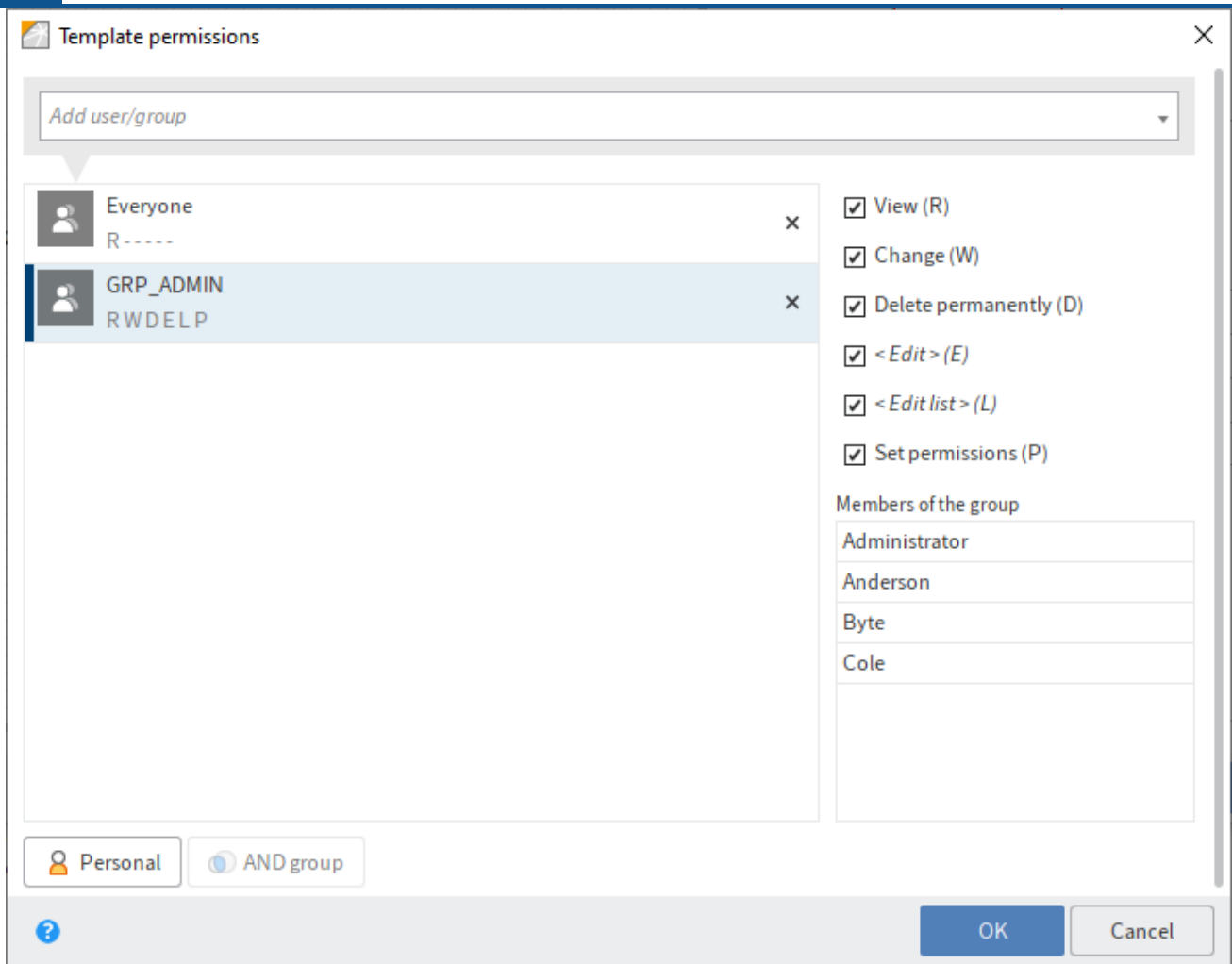
Delete version: *Delete version* allows you to assign a deletion marker (highlighted red) to the selected versions. The version can be deleted permanently via *Ribbon > Organize > System > Delete permanently*.

Restore: *Restore* allows you to remove the deletion marker from the selected version.

Permissions



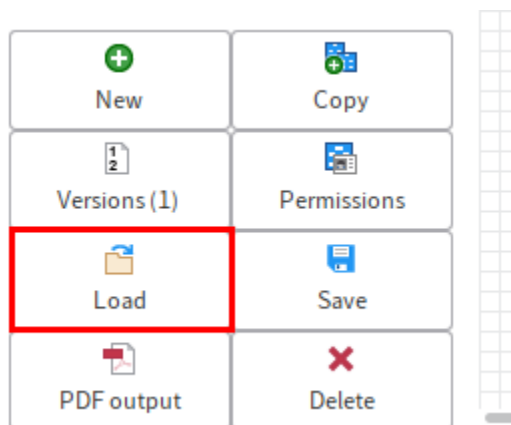
Once you have selected a workflow template and enabled edit mode, open the *Template permissions* dialog box via *Permissions*.



In the *Template permissions* dialog box, you specify the access rights to the workflow template.

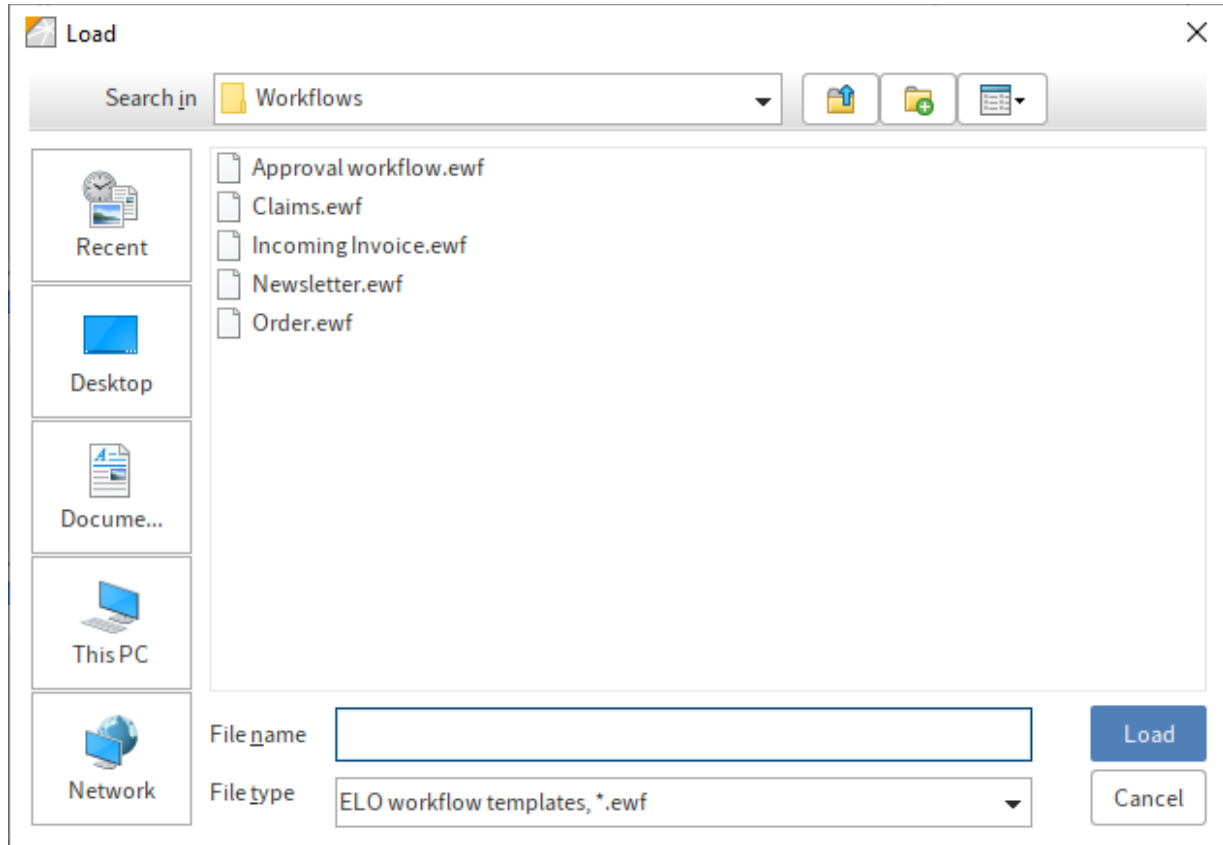
Load template

Use the *Load* function to import workflow templates. Workflow templates must always have the extension EWF.



1.

Select *Load*.



The *Load* dialog box opens.

Optional: If required, navigate to the location where the template is saved.

2. Select the desired template.
3. Select *Load*.

The selected workflow template appears in the workflow designer.

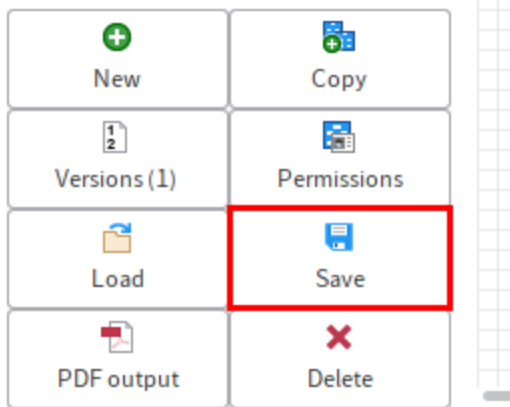
Save template

Use the *Save* function to export the selected workflow template as an EWF file.

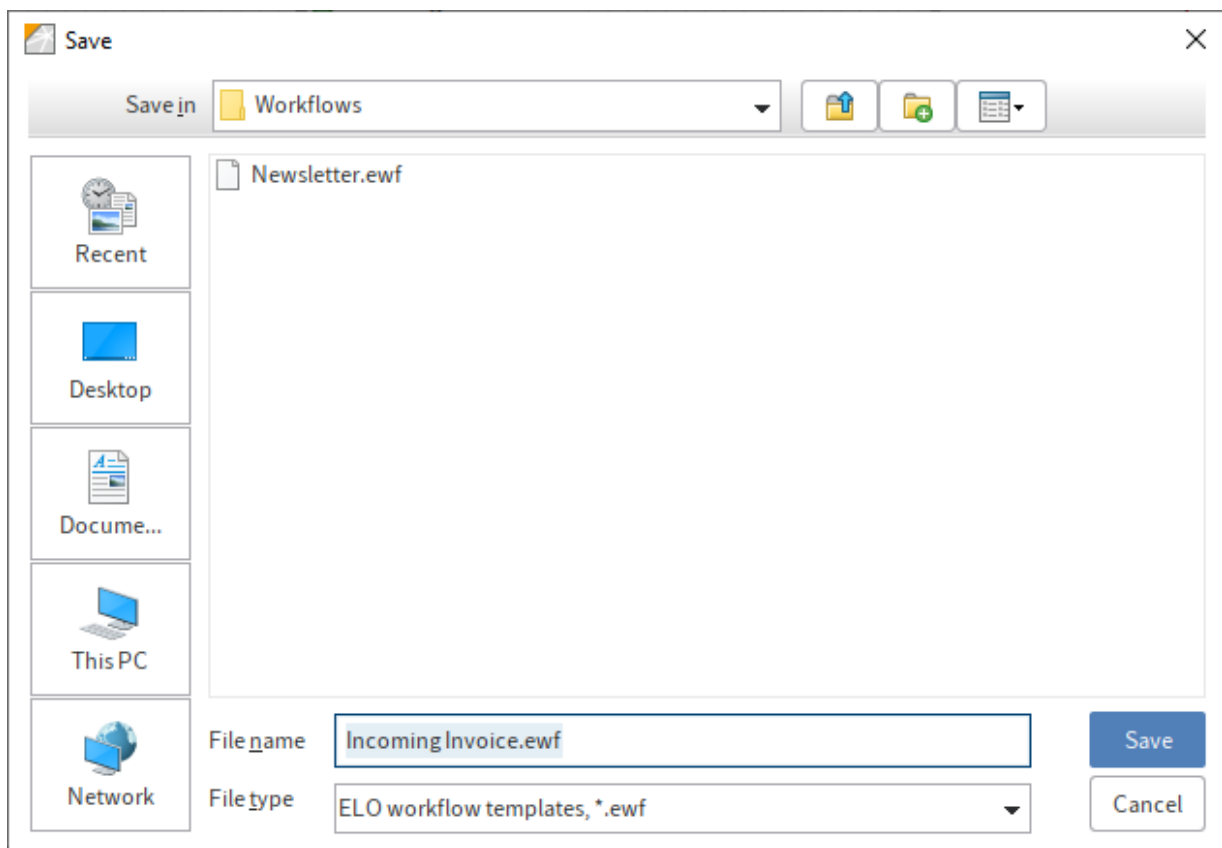
Information

To save the current status of a workflow template in ELO, create a new version or select *Apply*.

1. Select a workflow template in the *Templates* column.



2. Select *Save*.



The *Save* dialog box opens.

Optional: If required, select another storage location.

3. Enter a name for the workflow template.

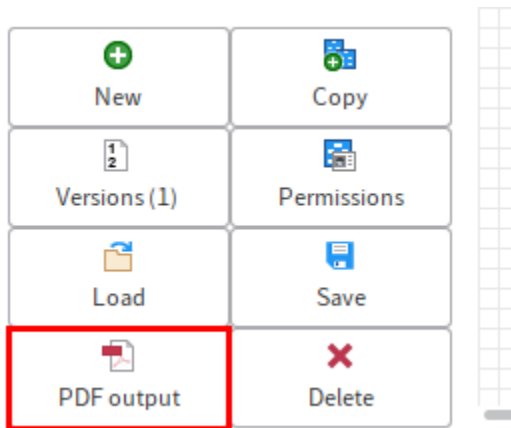
4. Select *Save*.

The workflow template is saved externally.

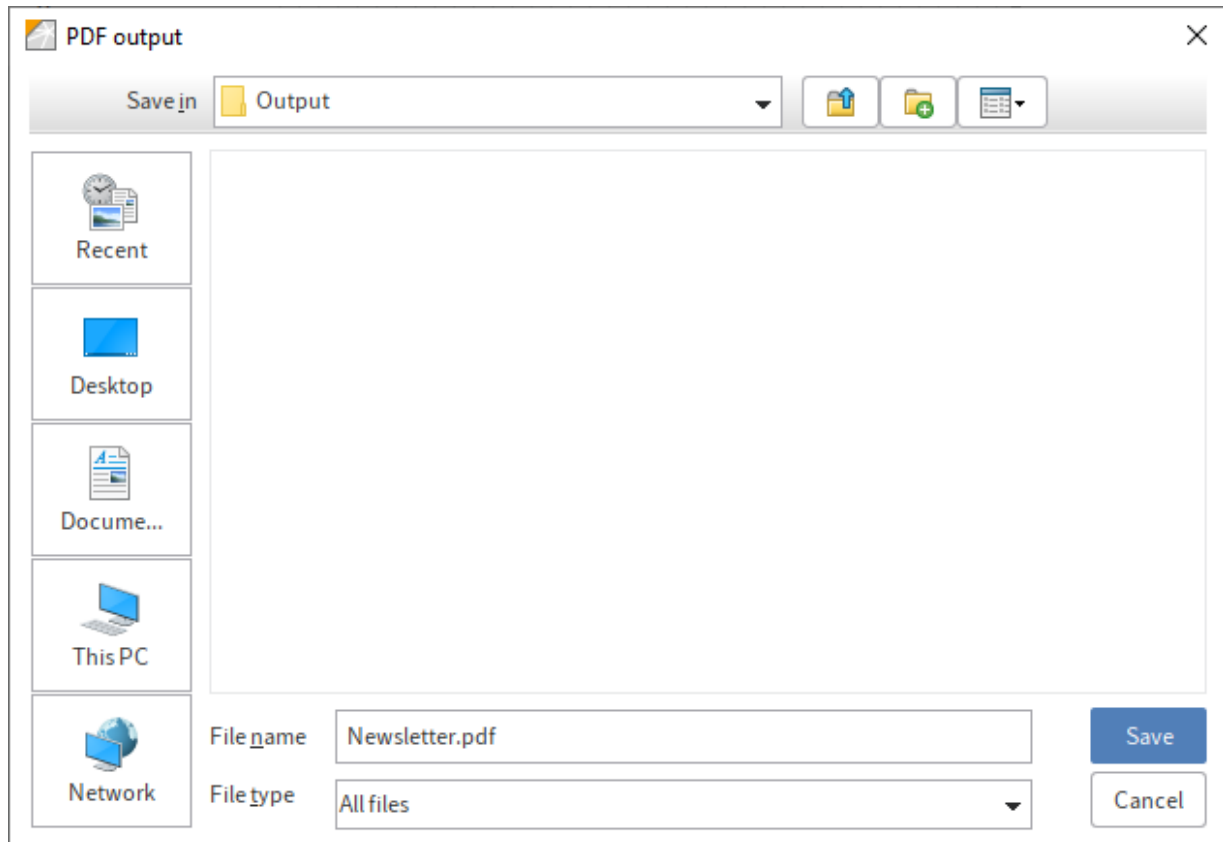
PDF output

The *PDF output* function creates an overview of the selected workflow template as a PDF file.

1. Select a workflow template in the *Templates* column.



2. Select *PDF output*.



The *PDF output* dialog box opens.

3. Select the location where you want to save the PDF file.

Optional: You can change the name of the file.

- 4.

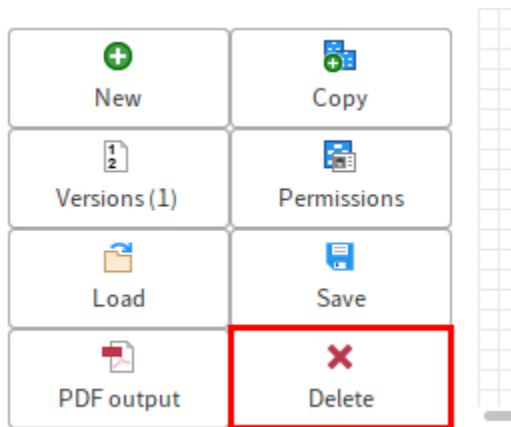
Select *Save*.

ELO creates a PDF file from the selected workflow template.

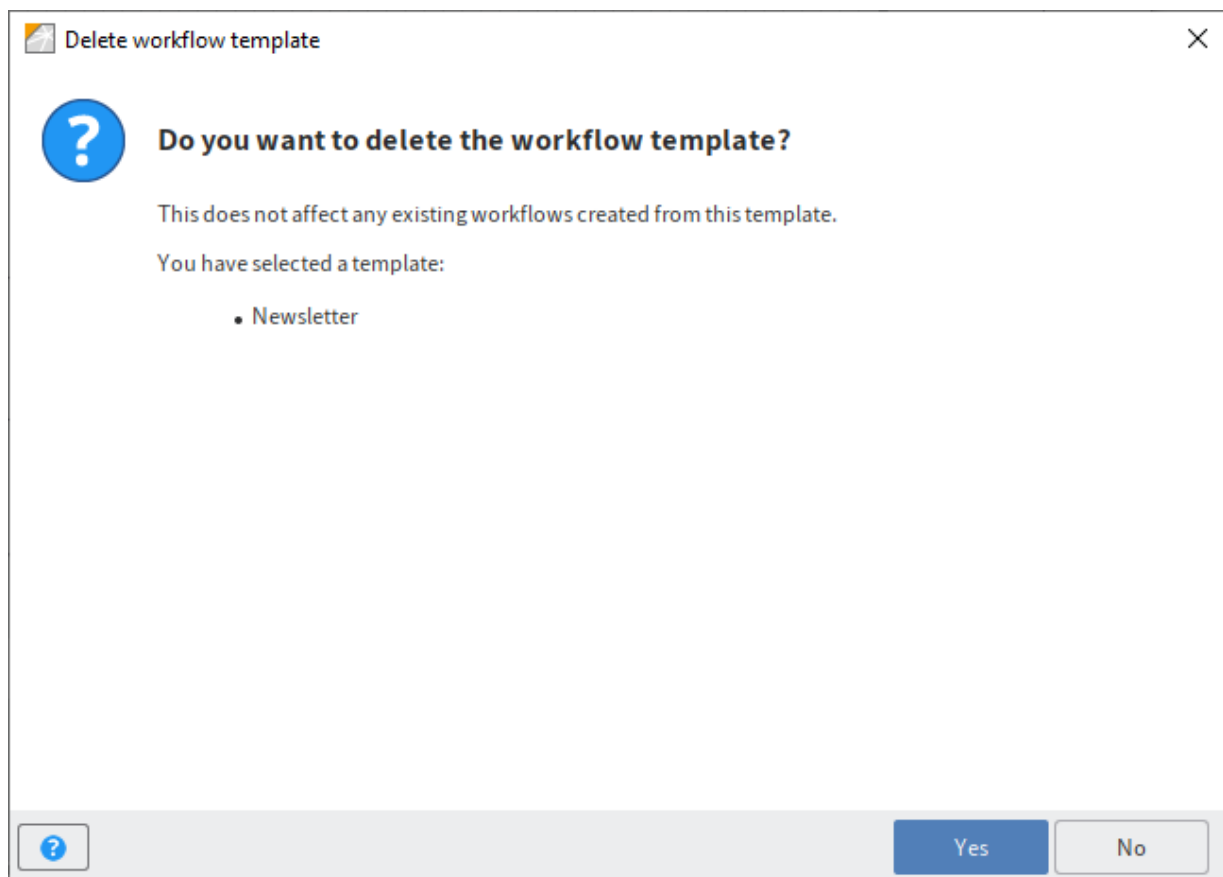
Delete template

Remove a workflow template from ELO as follows:

1. Select the workflow template that you want to delete in the *Templates* column.



2. Select *Delete*.



A confirmation prompt appears.

3. Click **Yes** to confirm the delete action.

The workflow template will be deleted and no longer appears in the *Templates* column.

Form-based workflow

Introduction

In ELO, forms can be used in different locations:

- Form for editing a workflow
- Form as metadata preview
- Form as a substitute for a metadata form
- Form for creating data sets in ELO for Mobile Devices

This chapter focuses on the interaction between workflows and forms.

Form and metadata

Each form must be connected to a metadata form. The information entered in the form is saved via the metadata form. The metadata creates the connection between the form and database.

All fields on a metadata form, the extra text, and map fields can be used as storage locations.

Map fields are freely definable fields whose contents are saved to the database. The contents of certain map fields can be seen in the metadata via the *Additional information* tab in the metadata when the user has the corresponding right to it.

Theoretically, you can generate an infinite number of map fields. The fields save information from dynamically created form fields since the map fields are also created dynamically. However, fields have certain features that are not available to map fields. For example, the conventional ELO search functions cannot be used to search the contents of map fields.

Add a separate metadata form for each form and align it with the fields used in the form.

Technology

In general, all forms in ELO are based on HTML, CSS, and JavaScript. ELO provides the forms via the *ELO Web Forms Services (ELOWf)* module.

ELO saves HTML, CSS, and JavaScript information of the forms in TXT documents. These TXT documents are stored in ELO at *Administration // ELOWf Base // Forms*.

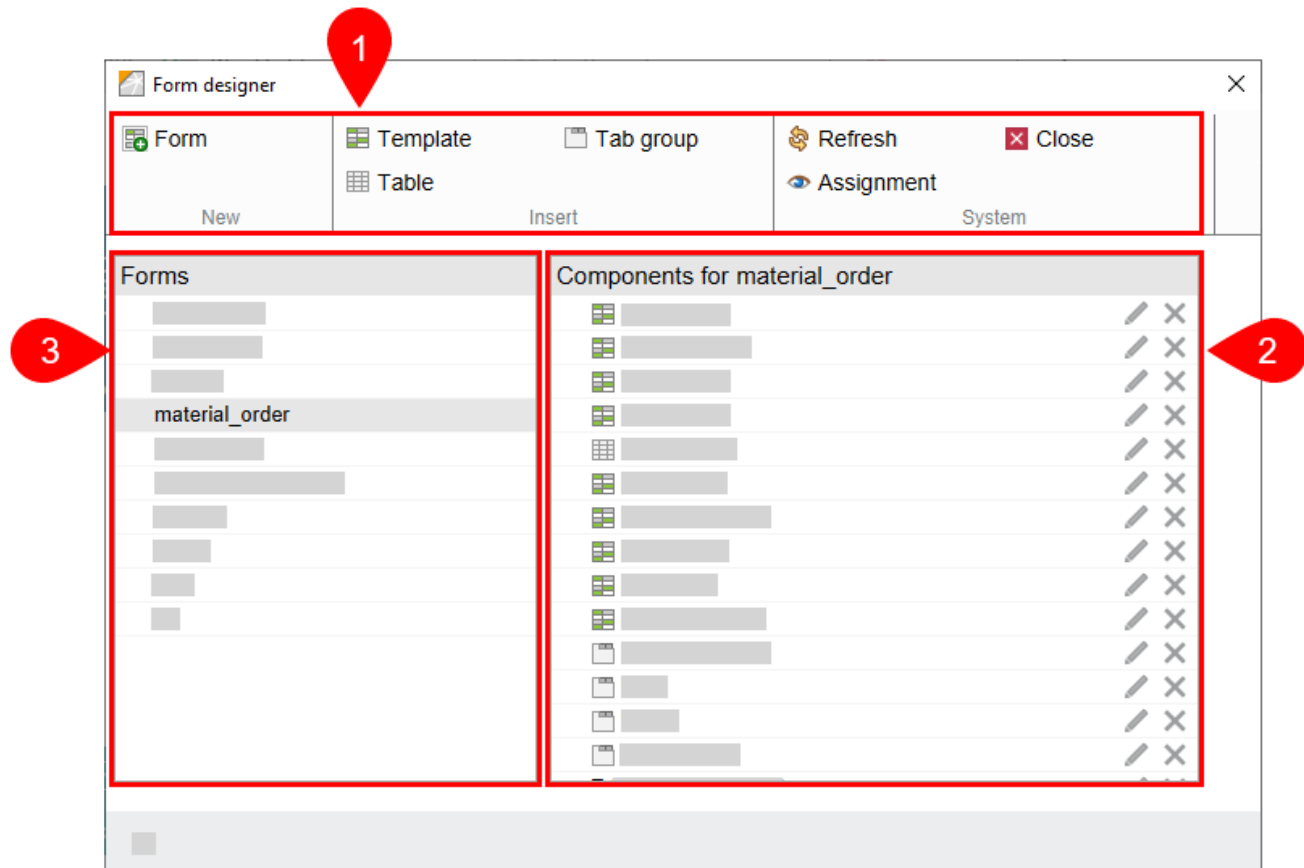
The ELO form designer is used for creating and editing forms. Open the ELO form designer in the ELO Java Client via *Ribbon > Organize > System > Form designer* or in the ELO Administration Console via *Server modules > Form designer*.

The form designer

The ELO form designer is used to create, edit, and manage ELO forms. Open the ELO form designer via *Ribbon > Organize > System > Form designer*.

Alternatively: You can also open the ELO form designer via the ELO Administration Console (*ELO Administration Console > Server modules > Form designer*).

The home screen of the ELO form designer is made up of the following areas:



1 **Toolbar:** The toolbar offers you the basic functions of the ELO form designer:

2 **'Components for' column:** You see all existing components in the *Components for* column. The components can be *Templates*, *Tables*, or *Tab groups*. Another component is the *Form header scripts*. This component is added automatically as soon as you create a form.

3 **'Forms' column:** You see all forms in ELO in the *Forms* column.

Toolbar

The toolbar gives you access to the following functions:

Form: Use this function to create a new form.

Template: Use this function to create a new template for a form. Templates are the building blocks for forms. A form can consist of multiple templates and/or tables.

Tab group: Use this function to create a new tab group. Forms can be split into different tabs. This makes it easier to structure large forms.

Table: Use this function to create a new table. Tables are a special type of template. With table templates, you have the option to record data in table form and save it to the database using special map fields.

Refresh: Use this function to refresh the form data.

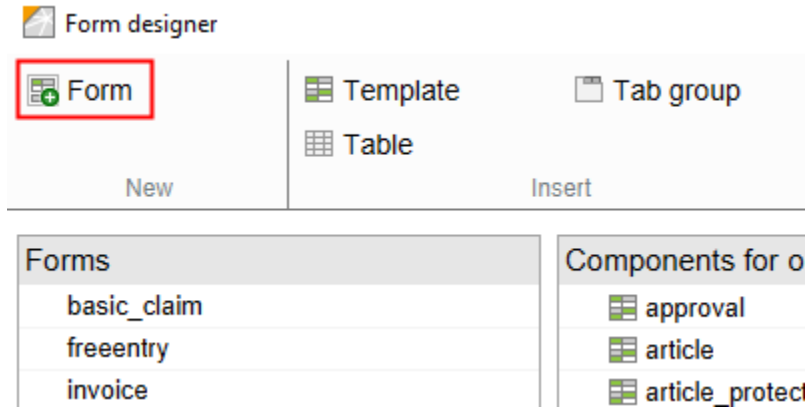
Close: Use this function to close the ELO form designer.

Assignment: Use this function to open a child menu. In this child menu, you can assign forms to individual metadata forms. These forms are displayed instead of the metadata preview if you select the corresponding option. You can edit the metadata via the form. If a form has not been assigned, ELO uses a default form.

Create form

You first need to set up a form grid to create form components. Proceed as follows:

1. Open the form designer.



2. Select *Form*.

Create new form

Enter the name of the new form.

You can only use letters and/or digits, and no special characters other than the underscore. The name can contain a maximum of 20 characters.

Name*

Metadata

The *Create new form* dialog box opens.

Optional: To create a form from a metadata form, select a metadata form from the *Metadata* drop-down menu.

3. Enter the name of the new form in the *Name* field.

The following rules apply for this:

- The first character must be a letter.
- No umlauts
- No special characters
- A minimum of one character
- Maximum 21 characters

Information

Due to technical reasons, ELO converts uppercase letters to lowercase.

4. Select *OK*.

The form appears in the *Forms* column. The entry *Edit form header scripts* appears under *Components for*. The basic data for the form is saved to ELO. You have now created the basic structure of the form.

You can add components to the new form. The type and arrangement of the components will depend on what you want to achieve with the form and respective workflow. The different components of a form are explained in the next section.

Components enable you to split a form into multiple parts. This way, you can control which parts of the form are displayed at which node and which parts can be edited.

Create templates

Templates are the building blocks of the ELO forms. ELO differentiates between three types of templates:

Standard templates: These templates, which are empty at first, can include a variety of field types.

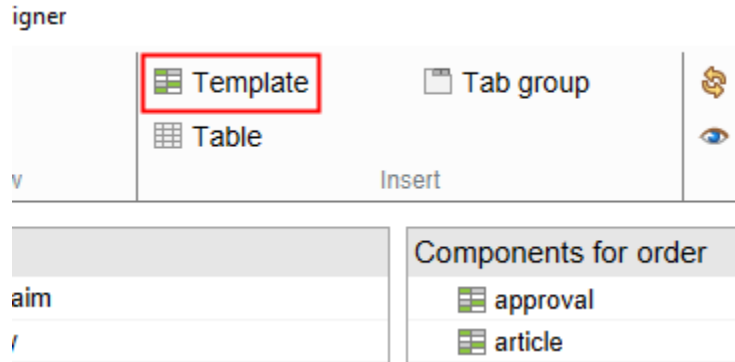
Table templates: The table data is saved automatically via special map fields. However, you need to follow special rules when adding tables. You can find additional information on tables in the Table attributes section.

Tab groups: Tab groups help to structure a form. These groups are only useful when at least two other templates exist. The tools available for tab groups differ from those for templates and tables. You can find additional information on tab groups in the Create tab groups section.

There is one basic method for creating templates, regardless of the type. In the following, we will demonstrate how to create a standard template.

Create the basic structure

1. Open the form designer.
2. Select the form that you want to add a template to.



3. Select *Template*.

Create new template

Enter the name of the new template.

You can only use letters and/or digits, and no special characters other than the underscore. The name can contain a maximum of 30 characters. Information: A period is used as a separator for mobile views and should therefore not be part of the name.

Name*

Copy from

The *Create new template* dialog box opens.

4. Enter a name for the new template in the *Name* field.

The same rules as for naming forms apply. See the *Create form* section.

Optional: Use the *Copy from* field to apply the settings of an existing template.

5. Select *OK*.

Form designer

New line	Delete line	Input	Check box	Image	JSAddLine	Delete	Save
New column	Delete column	Date	Radio button	Signature	JSRemoveLine	Cut	Apply
Merge	Split	Text	Combo box	Link		Copy	Save and preview
Table		Editor	Button	Relation		Paste	Cancel
				Insert	Edit		System

Damage report			
Customer data			
Insurance number	<input type="text"/>	Damage number	<input type="text"/>
Insured person	<input type="text"/>		
Last name	<input type="text"/>	First name	<input type="text"/>
Street	<input type="text"/>	Nr.	<input type="text"/>
Zip code	<input type="text"/>	City	<input type="text"/>
Damage			
	<input type="checkbox"/> Fire	<input type="checkbox"/> Hail	<input type="checkbox"/> Landslide
	<input type="checkbox"/> Flood	<input type="checkbox"/> Other	
Date of the accident	<input type="text"/>		

Properties of the selected cell

Field type:

Text:

Variable name:

Keyword list:

URL:

View type:

Tooltip:

Validation:

Validation message:

Formula:

Keyboard shortcut:

Character count:

Read-only

Form columns:

Global form settings

Metadata form:

Template name:

Languages:

Translation variable (prefix):

Limited variable access

Realign columns

Current cell contents

```
<div style="left: 0px; top: 0px;" elonodename="INPUT"><input size="15" name="IX_GRP_INSURANCENUMBER" value="" title="" eloverify="" accesskey="" type="text"></div>
```

The template is created. The form designer switches to a different mode.

You now see the following areas in the form designer:

- 1 **Toolbar:** The tools for creating and editing templates and tables.
- 2 **Properties of the selected cell:** This is where you edit the properties of the cell selected in the form area.
- 3 **Global form settings:** This is where you edit the settings that apply for the entire form.
- 4 **Current cell contents:** Shows the HTML structure for the cell selected in the form area.
- 5 **Form area:** The elements of the template or the table appear here.

Connect metadata forms/metadata

Each form must be connected to a metadata form with the metadata of an object in ELO. The time at which you create and integrate the metadata form depends on your preferred way of working.

Connect a form to a metadata form via the *Global form settings* area. You can make and change the settings via any template of a form. However, the settings always apply to the entire form.

Once you have created the metadata form, perform the following steps to connect it to the form:

Information

Newly created metadata forms do not appear immediately in the form designer. If required, restart the *ELO Indexserver* and then *ELO Web Forms Services*.

1. Open the form designer.
2. Open the desired form.
3. Open a template.

Global form settings

Metadata form	37: Damage report	▼
Template name	main_damagereport	
Languages		▼
Translation variable (prefix)		
	<input type="checkbox"/> Limited variable access	
	<input type="checkbox"/> Realign columns	

You see the current metadata form in the *Global form settings* area in the *Metadata form* field. The field can be edited even if it is grayed out.

- 1.

Select the arrow icon to the right of the *Metadata form* field.

The screenshot shows a 'Global form settings' dialog box. On the left, there are four fields: 'Metadata form', 'Template name', 'Languages', and 'Translation variable (prefix)'. The 'Metadata form' field is currently set to '0: Basic entry' and has a downward arrow icon to its right. This icon has been clicked, opening a dropdown menu. The dropdown menu lists the following options: 'Barcode recognition', 'Basic entry', 'Company', 'Damage report', 'Directive', 'Document', 'Documentation', 'ELOScripts', 'ELO user folder', 'E-mail', 'Folder', 'Form', 'Invoice', and 'Marketing'. Below the dropdown menu, there is a section labeled 'Current cell contents' which is currently empty.

A drop-down menu with available metadata forms appears.

2. Select the desired metadata form.

The selected metadata form is entered.

Design template

You can start designing the template once the basic structure has been created.

Use the form designer tools to do so. The tools are introduced in the *Toolbar* section. Also read the sections following this section.

Save

Once you have created the template and made all desired settings, save the template.

1. Select *Save*.

The form designer closes. The template now appears in the *Components for* column and can be integrated in a workflow, for example.

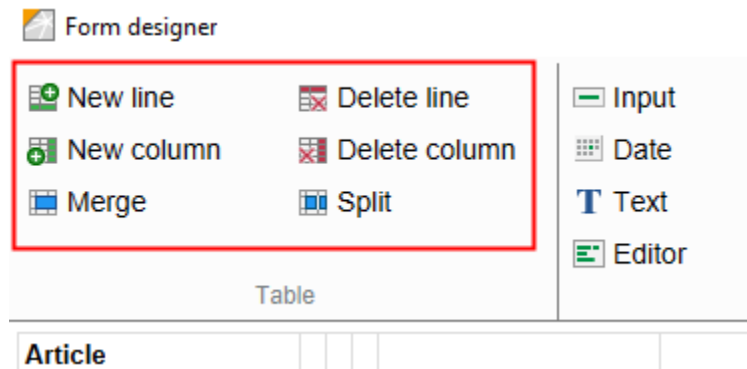
Information

If you want to save the template without closing the form designer, select *Apply* or *Save and preview*.

The toolbar

The different tools that appear in the toolbar when you create templates and tables are described below. The explanations for the toolbar for tab groups can be found in the Create tab groups section.

'Table' group



The layout of the templates in ELO is designed using a grid (as an HTML table). Edit this grid using the tools from the *Table* group.

New line

Use *New line* to add a new line to the grid. The new line appears below the currently selected line.

Delete line

Use *Delete line* to delete the currently selected line.

New column

Use *New column* to add a new column to the grid. The new column appears to the right of the currently selected column.

Delete column

Use *Delete column* to delete the currently selected column.

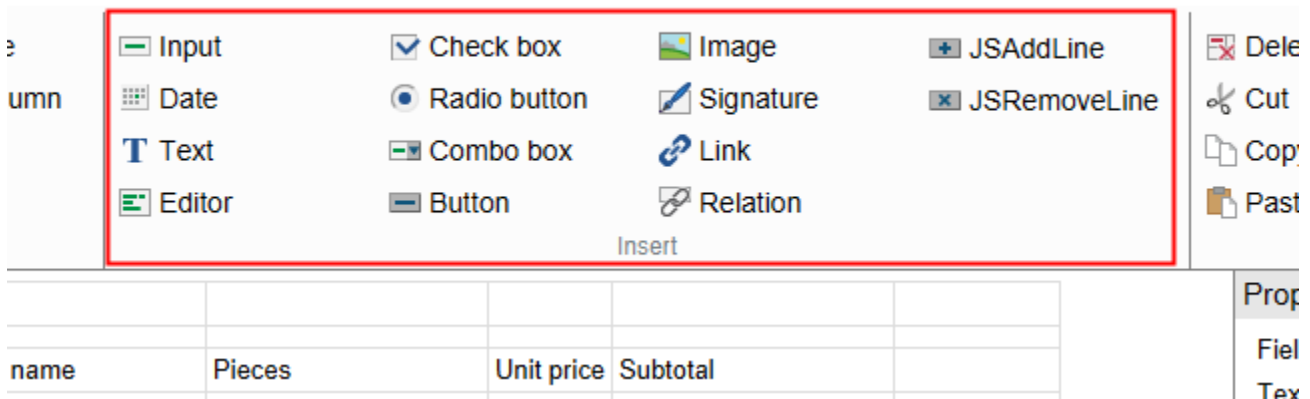
Merge

Use *Merge* to connect the selected cell with the cell to the right of it.

Split

Use *Split* to disconnect two cells.

'Insert' group



You can add different field types with the tools in the *Insert* group.

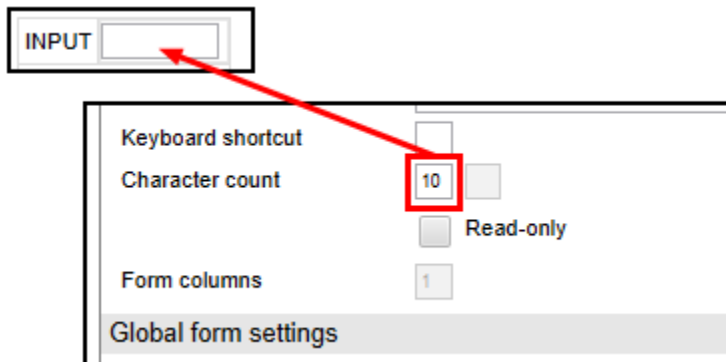
The field types vary in terms of function and setting options. The individual field types are briefly described in the following.

See also the *Cell properties* and *Validation* sections.

Input

Use *Input* to create an input field in the currently selected cell.

Users can enter text in the form via input fields. Input fields are restricted to one line.

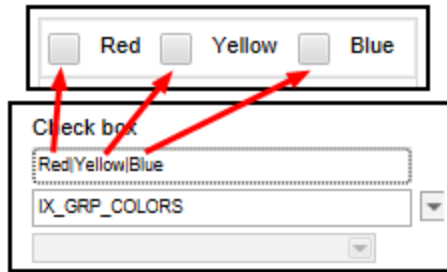


Use the value in the *Character count* field (in the *Properties of the selected cell* area) to determine the width of the input field.

Check box

Create a check box field in the currently selected line with the *Check box* button. A check box field can contain multiple check boxes. It is also possible to check multiple boxes.

Assign each check box in a check box field a name via the *Text* field (in the *Properties of the selected cell* area). There are two ways to do this:



A) Multiple check boxes in a check box field: Use a pipe symbol "|" to separate the names of the different check boxes. A new check box starts after every pipe symbol.

Assign the desired metadata field (or map field) to the cell via the *Variable name* field (in the *Properties of the selected cell* area). The selected field applies for all check boxes in the same cell.



B) Assign check boxes to multiple cells: If you want to assign connected check boxes to multiple check box fields, create one check box per check box field and then assign the same metadata field (or map field) to each check box field.

Editor field	<input type="text"/>
Date	<input type="text"/>
Colors	<input type="text"/>
More colors	<input type="text" value="1 3"/>

ELO assigns a value to each check box. You see the value of the current check box in the *Current cell contents* under *value*. ELO enters this value into the field (or map field) linked to the check box field when the user clicks a check box in the form. If multiple check boxes are selected, ELO enters the corresponding value into the corresponding field in the metadata sequentially and separates each of the values with a pipe symbol "|".

You can then query these values with a decision node or a script, for example.

Image

Use the *Image* button to create an image field in the currently selected cell.

Image fields display image files. Proceed as follows to insert an image into a template:

1. File the image file to ELO in *Administration // ELOwf Base // Images*.
2. Enter a short name.

Information

You need to enter the short name in the form later so that the form designer can find the image. (See step 9.)

3. Open the form designer.
4. Select the *Refresh* button.
5. Open the desired form.
6. Open the desired template.
7. Select the desired cell.
8. Select *Image*.

Properties of the selected cell	
Field type	Image
Text	add.png
Variable name	<input type="text"/>
Keyword list	<input type="text"/>
URL	<input type="text"/>
View type	<input type="text"/>
Tooltip	<input type="text"/>
Validation	<input type="text"/>

The default image appears in the cell. You see the short name *add.png* of the default image in the *Text* field under *Properties of the selected cell*.

- 9.

Enter the short name of the image filed at the beginning in the *Text* field.

Information

You do not have to enter the extension unless it is part of the short name.

The image is then shown.

JSAddLine

Use *JSAddLine* to create a button in the currently selected cell that the *JS_ADDLINE* variable has already been assigned to.

Information

You should not add more than 100 lines with the function *JS_ADDLINE*. Otherwise, the performance of the forms will be impaired.

There is a limit of 2000 characters.

Properties of the selected cell

Field type	Button
Text	<input data-bbox="527 1018 967 1060" type="text" value="+"/>
Variable name	<input data-bbox="527 1066 967 1108" type="text" value="JS_ADDLINE"/> ▼
Keyword list	<input data-bbox="527 1115 924 1157" type="text"/> ▼
URL	<input data-bbox="527 1163 967 1205" type="text"/>

Buttons with the variable *JS_ADDLINE* give users the option to duplicate the line above the button when filling out the form.

You can change the name of the button in the *Text* field.

lines

To duplicate multiple lines, enter the parameter *lines:* and the number of lines you need in the *Validation* field. To duplicate the three lines above, for example, enter `Lines:3`.

max

The attribute *max* in the validation field of the form designer specifies the maximum number of lines that can be added. Once this number is exceeded, the button will be disabled. If more data is available in the database when the form is loaded, all data is loaded to the form even if the maximum parameter has been exceeded.

addlineid

If you have multiple JS_AddLine fields on a form, you can increase them using an *addlineid* in the validation field of the form designer. This ID is used in scripting functions to distinguish between the JS_AddLine fields.

Date

Use the *Date* button to create a date field in the currently selected cell.

Users can enter a date via date fields. A calendar icon appears next to the date field. Use the calendar icon to open a calendar and select the date.

Use the value in the *Character count* field (in the *Properties of the selected cell* area) to determine the width of the date field.

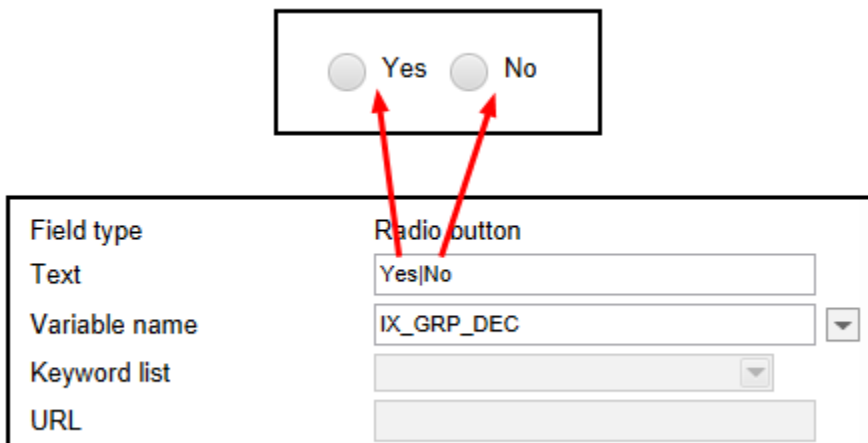
The *date* value is automatically entered in the the *Validation* field (in the *Properties of the selected cell* area). This ensures that only date entries can be made.

For more information on validation, refer to the Validation section.

Radio button

Use *Radio button* to create a radio button field in the currently selected line. A radio button field can contain multiple radio buttons. Only one option can be selected for connected radio buttons.

There are two ways to create radio buttons fields:



A) Multiple radio buttons in a radio button field: Use a pipe symbol "|" to separate the names of the different radio buttons. A new radio button starts after every pipe symbol.

Assign the desired field (or map field) to the cell via the *Variable name* field (in the *Properties of the selected cell* area). The selected field applies for all radio buttons in the same cell.

B) Distribute radio buttons to multiple cells: If you want to distribute connected radio buttons to multiple cells, add one radio button field per cell and then assign it the cell the same field (or map field).

Depending on which option the user selects, the respective value is saved to the corresponding field (or map field).

Signature

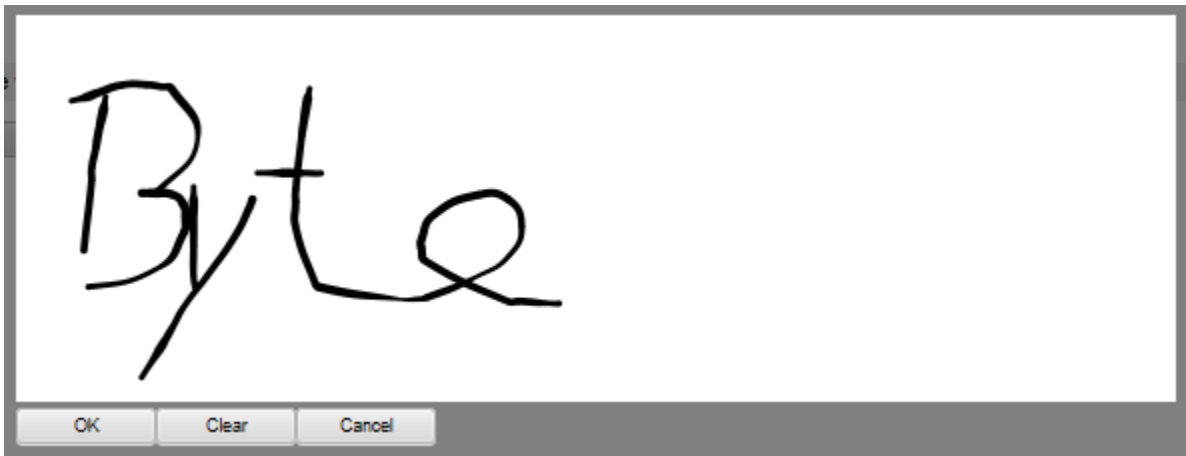
Use the *Signature* button to create a signature field in the currently selected cell. A signature field is used to link signatures to a form.

Properties of the selected cell	
Field type	Signature
Text	Sign
Variable name	IX_BLOB_WO_TEST
Keyword list	
URL	
View type	

For the signature field to work, the variable that you enter in the *Variable name* field (in the *Properties of the selected cell* area) must follow this pattern:

IX_BLOB_WO_<NAME>

- BLOB: Binary Large Object
- WO: write once



A signature field initially appears as a button that you can open a drawing area with.

You can draw a signature in this space. There are various methods for doing this:

- Mouse: Hold down the left mouse button and write your signature.
- Touch screen device: Sign with your finger.
- Stylus pen: Write your signature with a stylus pen.



Click *OK* to save the signature.

Information

The signature will not be linked to the form until you have saved the form or forwarded the workflow.

The signature is shown in the form with a timestamp.

JSRemoveLine

Use the *JSRemoveLine* button to create a button in the currently selected cell that the *JS_REMOVELINE* variable has already been assigned to. A button with the variable *JS_REMOVELINE* is used to delete duplicated lines. This deletes the row in which the button is located.

Article	Article number	Pieces	Unit price	Subtotal	
	<input type="text" value="BM-87-V6150"/>	<input type="text" value="10"/>	<input type="text" value="1"/>	<input type="text" value="10"/>	<input type="button" value="X"/>
	<input type="text" value="BM-87-C1293"/>	<input type="text" value="10"/>	<input type="text" value="2"/>	<input type="text" value="30"/>	<input type="button" value="X"/>
	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text" value="0"/>	<input type="button" value="X"/>
<input type="button" value="Add article"/>					
Total				<input type="text" value="30"/>	

This button always appears on the form as an X icon. The names of *JS_REMOVELINE* buttons cannot be changed.

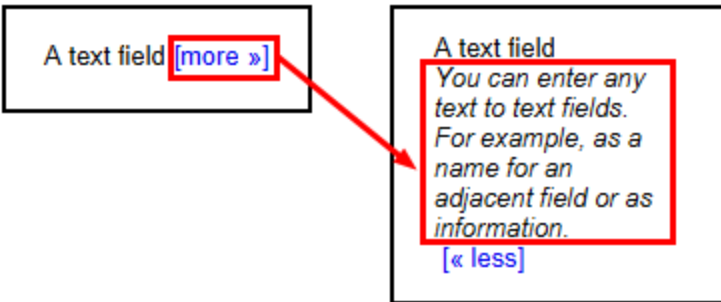
Information

To delete multiple lines, enter the parameter *lines:* and the number of lines you need in the *Validation* field. To delete the three lines above, for example, enter *lines:3*.

Text

Use the *Text* button to create a text field in the currently selected cell. If you enter text in an empty cell, the cell automatically turns into a text field.

You can enter any text in text fields. For example, as a name for an adjacent field or information that is permanently displayed.




Text fields also enable users to add explanatory text. This text is initially hidden. Instead, the linked text *[more >>]* appears. The explanatory text is displayed when you click the link. Clicking *[<< less]* hides the explanatory text.

Add explanatory text to a text field as follows:

1. Select a text field

Properties of the selected cell

Field type	Text
Text	A text field 
Variable name	<input type="text"/>
Keyword list	<input type="text"/>
URL	<input type="text"/>
View type	<input type="text"/>

2. Select the button next to the *Text* field (in the *Properties of the selected cell* area).

Explanatory text

Enter the explanatory text, which is displayed after clicking [more...].

You can enter any text in text fields. For example, as a name for an adjacent field or as information.

OK Cancel

The *Explanatory text* dialog box opens.

3. Enter the desired text.
4. Select *OK*.

The text is saved. The dialog box closes. The *[more>>]* text appears after the main text in the text field.

Combo box

Use *Combo box* to create a combo box field in the currently selected cell.

Combo box fields provide a list of terms that can be selected. The respective selected term is saved to the associated field (or map field).

Properties of the selected cell

Field type	Combo box
Text	Apple Pears Plums Cherries Mirabelles
Variable name	IX_MAP_FRUIT
Keyword list	
URL	
View type	

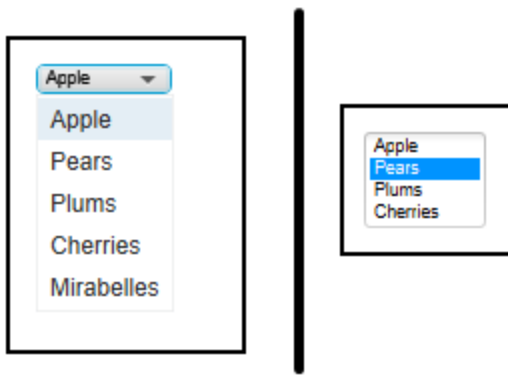
Add list entries via the *Text* field (in the *Properties of the selected cell* area). The individual entries are separated by a pipe symbol "|".

Information

Alternatively, you can use keyword lists and user lists. For additional information about this function, refer to the section on *Cell properties*.

Please note

You cannot use dynamic keyword lists in combo box fields.



Use the *Character count* field to determine the maximum number of lines that will be displayed in the combo box field. If the value in the *Character count* field is 1, only the first entry is displayed. All additional entries can be selected via a drop-down menu. If the value is greater than 1, the respective number of entries is displayed. In addition, a scroll bar appears on the side if all available entries cannot be displayed.

Link

Use *Link* to create a link field in the currently selected cell.

Use link fields to embed links in websites and documents in forms. For websites, enter the corresponding URL in the *URL* field. In the case of documents, enter the ELO GUID of the document (including brackets) in the *URL* field.

Please note

If you do not enter a link text in the *Text* field, the link will not be displayed.

Properties of the selected cell

Field type	Link
Text	<input type="text" value="Office buildings"/>
Variable name	<input type="text"/>
Keyword list	<input type="text"/>
URL	<input type="text" value="(EF68E78C-E1CB-4CBD-8583-0C4F48304'"/>
View type	<input type="text"/> ▼
Tooltip	<input type="text"/>
Validation	<input type="text"/> ▼

A left-click on a link to a document opens the external standard browser and downloads the document.

Right-click the link to open a context menu. You have the following options in the context menu:

-

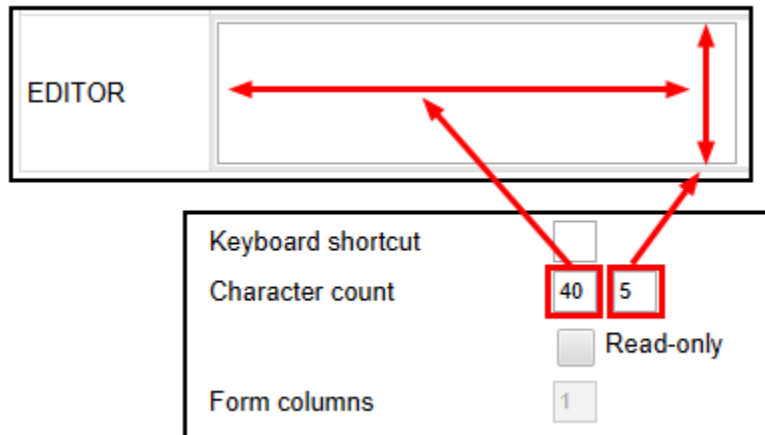
Open link: Opens websites in the ELO browser. This does not work for documents.

- Open link in new window: Opens websites in the ELO browser. This does not work for documents.
- Copy link to clipboard: Copies the link to the document or website to the Windows clipboard from where it can be moved to other locations.

Editor

Use *Editor* to create an editor field in the currently selected cell.

The user can enter large text blocks in the form via editor fields.



Use the value in the *Character count* field (in the *Properties of the selected cell* area) to determine the width of the editor field.

Use the value next to the *Character count* field to determine the height of the editor field.

Button

Use the *Button* function to create a button in the currently selected line that can be linked to your own JavaScript function.

Properties of the selected cell	
Field type	Button
Text	Determine days
Variable name	JS_DAYS
Keyword list	
URL	
View type	

Own script functions must be named according to the following pattern:

JS_<NAME>

In addition, these script functions must be stored in the *Edit form header scripts* component.

Relation

You can use the *Relation* function to create a field in the currently selected cell that retrieves metadata from another entry.

Short name	Sector
Green's	Garden Supplies
Renzum Co.	Finance
WeKraTex	Textiles
Dan's Disposal Services	Disposal and recycling
Freddy's Forestry Services	Forestry
Smith's	Trades
SQL Pros	Software
Vinny's Vacuums	Retail

A form field of the type *Relation* is linked to a metadata field of the type *Relation*. Fields of the type *Relation* are linked to one or more metadata forms that can be used for relations. This enables you to transfer metadata from one entry to another.

An automatically generated keyword list is provided for *Relation* type fields. It consists of the short names of the entries that are linked to the field through the corresponding metadata forms. In addition, up to five priority fields can be displayed in the keyword list.

Information

For more information on creating [Relation type fields and corresponding metadata forms](#), refer to the section [Create metadata form > Usage](#) in the documentation [Metadata forms and fields](#).

Properties of the selected cell

Field type	Relation
Text	<input type="text"/>
Variable name	IX_GRP_COMPANY <input type="button" value="v"/>
Keyword list	Keyword <input type="button" value="v"/>
Group name	COMPANY <input type="button" value="v"/>
	<input type="checkbox"/> Only list values allowed
URL	<input type="text"/>
View type	<input type="text"/> <input type="button" value="v"/>
Tooltip	<input type="text"/>
Validation	relation <input type="button" value="v"/>

For a *Relation* type field to work, you need to enter the corresponding *Relation* metadata field type in the *Variable name* field under *Properties of the selected cell*.

You need to enter the group name of the field in the metadata under *Keyword list*. You do not need to assign a keyword list to the field. This is generated automatically.

The validation value must be *relation*.

'Edit' group

dLine	<input type="button" value="Delete"/>	<input type="button" value="Save"/>
moveLine	<input type="button" value="Cut"/>	<input type="button" value="Apply"/>
	<input type="button" value="Copy"/>	<input type="button" value="Save and preview"/>
	<input type="button" value="Paste"/>	<input type="button" value="Cancel"/>
	Edit	System

Properties of the selected cell

The *Edit* group contains tools for editing cells.

Delete

Delete the contents of the selected cell via *Delete*.

Cut

Cut out the contents of the selected cell via *Cut*. Use *Paste* to insert the copied contents into another cell.

Information

The contents of a cell can also be dragged and dropped onto another cell.

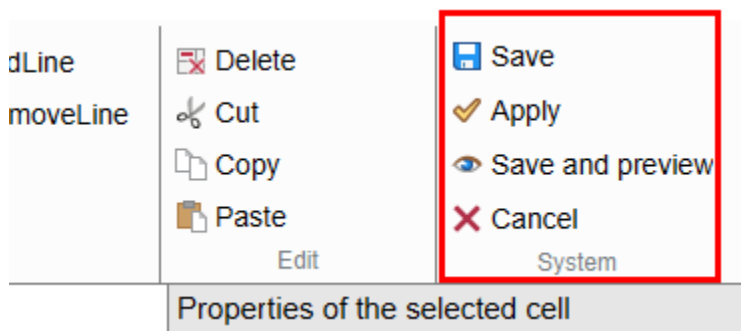
Copy

Use *Copy* to copy the contents of the selected cell. Use *Paste* to insert the copied contents into another cell.

Insert

Use *Paste* to insert the contents of a previously cut out or copied line.

'System' group



You will find buttons that apply for the entire template in the *System* group.

Save

Use *Save* to save all changes and close the template.

Apply

Use *Apply* to save all changes. The template remains open.

Save and preview

Use *Save and preview* to save all changes. The form also opens in the *ELO browser* dialog box. In the *ELO browser* dialog box, you can check the layout of the form and test some functions. You can also open the ELO debugger from the context menu of the ELO browser.

Please note

Not all scripts and functions will necessarily behave as they do in the actual environment that the form is being used in. You should therefore test the form in an appropriate environment before using it.

Use *Close* to close the dialog box and go back to the form designer.

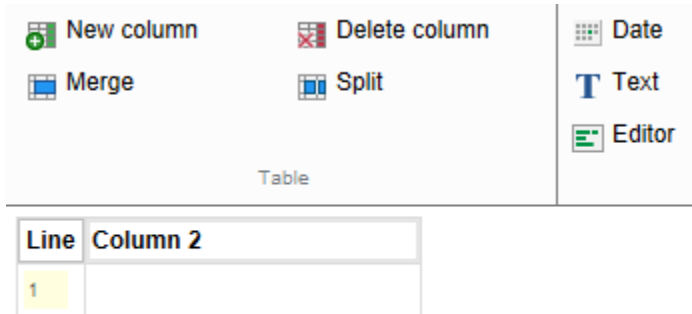
Cancel

Use *Cancel* to close the template without saving the changes. If there are unsaved changes, you need to confirm that you want to discard them.

Table attributes

Tables are a special type of template. Table templates consist of dynamically generated tables. Table data is saved automatically via special map fields.

Table templates are generally added the same way as standard templates. The same tools are available for this. However, there are a few things that you need to note in terms of their functionality and design.



Tables are dynamic components. By default, they contain a header and a data line, which you can modify as required.

Table header

The first line is automatically the table header and is therefore formatted with the *tbfirstrow* class. The field type is a *text field* by default. Change the name of the column in the *Text* field (in the *Properties of the selected cell* area). In addition, all other field types are allowed in the table header. However, they are not formatted as a table header.

First column

The first column is for counting lines automatically and must not be changed. It is formatted with the *tbfirstcol* class.

Add columns

Use the *New column* button to add columns to the table. The column header is automatically assigned the *tbfirstrow* class as soon as you enter text.

Data line

Each table only requires one data line. All additional lines are added dynamically when completing the form. Once the user completes the first line, another line is inserted at the bottom of the table.

Fields

Data cells may only include input fields so that tables work as intended. Keyword lists can be stored for the input fields.

Data storage

Line	Item	Item number	Piece	Unit price
1	Recycled paper, 500 sheets	8343	10	3.29
2	Office chair "Paris", midnight blue	7399	5	399.99
3	Eco-friendly mouse	5612	5	31.90
4				

Metadata	
Basic	
Extra text	
Options	
Permissions	
Version history	
Additional information	
Name	Value
ITEM_1	Recycled paper, 500 sheets 8343 10 3.29
ITEM_2	Office chair "Paris", midnight blue 7399 5 399.99
ITEM_3	Eco-friendly mouse 5612 5 31.90

A specific attribute of tables is that data is principally saved via map fields.

Global form settings

Metadata form	8: Order
Template name	article_table
Map name	ARTICLE
Languages	
Translation variable (prefix)	
	<input type="checkbox"/> Limited variable access
	<input checked="" type="checkbox"/> Realign columns

The *Map name* field in the *Global form settings* area is used for this. Enter a name here that you want the data to be saved under. ELO automatically numbers the map fields in ascending order. Each line is assigned a number. The column contents are separated by a pipe symbol in the metadata.

Please note

The map name must not contain special characters or spaces.

Create tab group

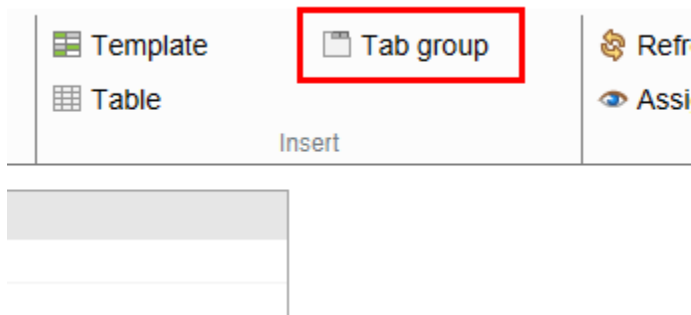
Tab groups are a special type of template. Tab groups are used to structure forms. A form can be split into multiple tabs. The tabs can also be broken down into subtabs. Each tab can include multiple templates, tables, and/or tab groups.

Information

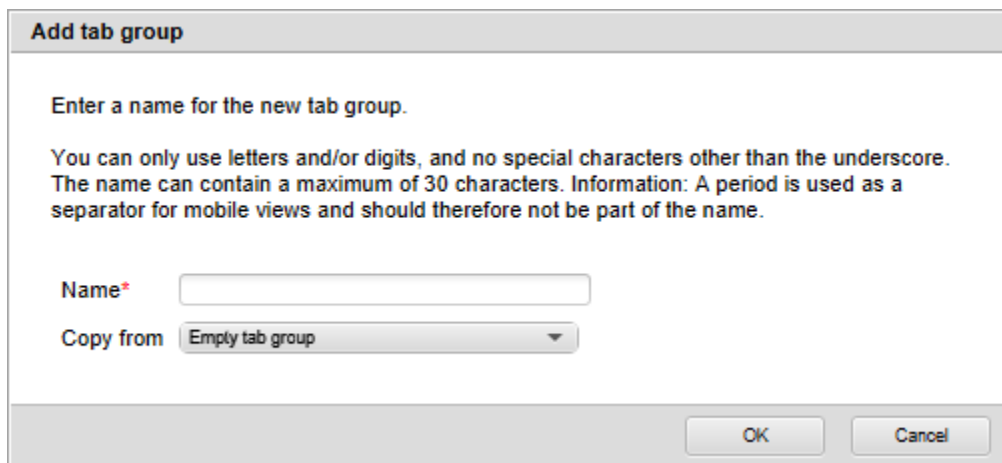
Data set forms for ELO for Mobile Devices must not be divided into tab groups. A data set form must be created as a separate template.

Tab groups are generally added the same way as standard templates. The design and functionality of tab groups, however, differs from the other templates. The following section describes how to create tab groups and includes examples of different configurations.

1. Open the form designer.
2. Select the form that you want to add a tab group to.



3. Select *Tab group*.

A dialog box titled 'Add tab group'. It contains the following text: 'Enter a name for the new tab group.' followed by 'You can only use letters and/or digits, and no special characters other than the underscore. The name can contain a maximum of 30 characters. Information: A period is used as a separator for mobile views and should therefore not be part of the name.' Below this text is a text input field labeled 'Name*' and a dropdown menu labeled 'Copy from' with 'Empty tab group' selected. At the bottom right, there are 'OK' and 'Cancel' buttons.

The *Add tab group* dialog box opens.

4. Enter a name for the tab group in the *Name* field.

The same rules as for naming forms apply (See the *Create form* section).

Optional: Use the *Copy from* field to apply the settings of an existing template.

5. Select *OK*.

The tab group is added. Then, the form designer switches to *Edit tab group mode*.

6. Configure the tab group according to your requirements. Keep the following notes in mind:

Tab ID: Assign a unique ID for each tab group. The ID is important when you are using scripts, for example.

Header: Enter a name for the respective tab here. Once you have entered a name and switch to another field, an additional editing line for further tabs appears.

Add component: Select one or more components (templates, tables, or other tab groups) via the drop-down menu in the *Add component* column. Multiple components can be placed on a tab.

The selected components appear in the column next to *Add component*. The list in this column can be edited as follows:

- Arrow icons: Use the arrow icons to change the display order of the components. The topmost component in the list also appears at the top in the respective tab.
- X icon: Use the X icon to remove the selected component from the respective tab.

Start element: Use the drop-down menu in the *Start element* column to select a variable, if required. The variable stands for a form field. When editing a form, the chosen form field is automatically selected when the respective tab is opened. If you do not select a variable, a field will not be automatically selected.

Example 1: Simple tab group

Form designer

Edit tab group

Tab ID

Header	Add component	Start element
Order	[Select] → formheader orderedby article	↑ ↓ × IX_GRP_ORDER
Approval	[Select] → formheader approval article_prot	↑ ↓ ×
Conclusion	[Select] → formheader orderby_prot article_prot approval_prot conclusion	↑ ↓ ×
	[Select] →	↑ ↓ ×

OK Cancel

In the example above, the three tabs *Order*, *Approval*, and *Conclusion* were added. The form looks like this:






























Order Approval Conclusion

Material order form




On the selected *Approval* tab, the *approval* template appears in the top part of the form. The *items_protection* template appears in the lower part. It is a copy of the *items* template that the user can place an order with. The copy is read-only (designated by *_protection*) and serves exclusively for displaying the entered values.

Example 2: Tab group with subgroup

















You have the option of creating subgroups for a tab group. The following example illustrates this:

Components for multitab	
 level1	 
 level2_1	 
 level2_2	 
 table1	 
 table2	 
 template1	 
 template2	 
 template3	 
 template4	 
Edit form header scripts	 

First, we created several templates and table templates. We also created the tab groups *level1*, *level2_1*, and *level2_2*. These are initially empty.

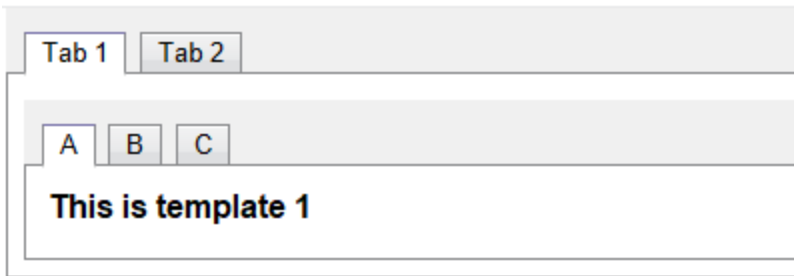
Tabs	
Header	Add component
Tab1	[Select]  → level2_1
Tab2	[Select]  → level2_2
	[Select]  →

We added *Tab1* and *Tab2* to the tab group *level1*. The first tab is connected to the tab group *level2_1*. The second tab was connected to the tab group *level2_2*.

Tabs			Start eler
Header	Add component		
A	[Select]  → template1	  	
B	[Select]  → template2 table2	  	
C	[Select]  → template3	  	
	[Select]  →	  	

Some tabs were added to the tab groups *level2_1* and *level2_2*. The tabs were assigned components.

Multitab



The image shows a multitab form structure. At the top, there are two tabs labeled "Tab 1" and "Tab 2". Below these tabs, there is a section containing three sub-tabs labeled "A", "B", and "C". Underneath these sub-tabs, the text "This is template 1" is displayed. The entire structure is enclosed in a light gray border.

The form now has the desired structure.

Cell properties

The following setting options are located in the *Properties of the selected cell* area.

Field type

When you have selected a cell with a form field, you see the field type under *Field type*.

Text

The *Text* field has different tasks depending on the selected field type. For example, it is used to label text fields and buttons or to select an image.

For more information, refer to the *Toolbar* section.

Variable name

The *Variable name* field serves various purposes:

- Creating a connection to a field.
- Specifying an action for a button.
- Entering variables for automatically completing fields.

Properties of the selected cell

Field type	Date
Text	<input type="text"/>
Variable name	<input type="text" value="IX_GRP_ORDER"/> ▼
Keyword list	#DATE# ▼
URL	<input type="text"/>
View type	<input type="text"/> ▼

Create connection to a field: Use the drop-down menu for the *Variable name* field to get a list of available fields. The selection of fields depends on the metadata form connected to the form. You can see which metadata form is connected to the form in the *Metadata form* field, which is located under *Global form settings*. You can change the metadata form here, if required.

You also have the option of entering the variable directly in the *Variable name* field. You must use the following format:

```
IX_GRP_<GROUP NAME OF THE FIELD>
```

Properties of the selected cell

Field type	Button
Text	X
Variable name	JS_REMOVELINE
Keyword list	
URL	
View type	

Specifying an action for a button field: When you use a *button* field type, enter the name of the function to the *variable name* field that should be called up by the button.

The function must correspond to the following naming convention:

```
JS_<<NAME OF THE FUNCTION>
```

In addition, you need to store the corresponding function via the *Edit form header scripts* component.

There are also default functions such as *JS_ADDLINE*. Buttons with the variable *JS_ADDLINE* give users the option to duplicate the line above the button when filling out the form. A button with the variable *JS_REMOVELINE* is used to delete duplicated lines. This deletes the row in which the button is located.

Properties of the selected cell

Field type	Input
Text	
Variable name	WF_OWNER
Keyword list	
URL	
View type	
Tooltip	

Variables for automatically completing fields: There are default variables. These variables can be used to automatically insert content into fields. They can also be used by scripts for evaluation purposes.

Information

If you enter the variables described here in the *Variable name* field, only the corresponding contents are displayed in the form. When you save the value and/or want to process it, the variable must be read by a script. Write the value to a corresponding form field via the script.

Here is a list of the default variables:

Variable	Effect
WF_SINGLESELECT	If TRUE is returned, the workflow can only be forwarded to one successor node. If FALSE is returned, there can be multiple successor nodes. The output value depends on the properties of the respective node.
WF_OWNER	The name of the ELO user who started the workflow.
WF_OWNERID	The ID of the user who started the workflow.
WF_NAME	The name of the current workflow.
WF_NODENAME	The name of the current node.
WF_TEMPLATE	The name of the workflow template used.
WF_NODEOWNER	The name of the user who is editing the current node.
WF_NODEOWNERID	The ID of the user who is editing the current node.
NEXT_1 (NEXT_2, etc.)	The ID and the name of the next successor node with the lowest (or second-lowest) ID. For example 2 check.

Additional variables:

Variable	Effect
ELO_CONNECTUSERNAME	The name of the ELO user currently logged on.
ELO_CONNECTUSERID	The ID of the ELO user currently logged on.
ELOAS_PATH	The URL of the ELOas used.
ELO_SERVICEUSERID	The ID of the ELOwf service user currently logged on.

The following variables refer to the settings that were made for the current user in the ELO user manager:

Variable	Effect
ELO_USERPROP1	The Windows user name of the current user if the user is configured in the ELO user manager.
ELO_USERPROP2	The e-mail address of the current user if the user is defined in the user manager.
ELO_USERPROP3	The contents of the <i>Property 5</i> field.
ELO_USERPROP4	The contents of the <i>Action</i> field.
ELO_USERPROP5	The contents of the <i>Property 1</i> field.
ELO_USERPROP6	The contents of the <i>Property 2</i> field.
ELO_USERPROP7	The contents of the <i>Property 3</i> field.
ELO_USERPROP8	The contents of the <i>Property 4</i> field.
ELO_SUPERIOR	The ID of the current user's supervisor.
ELO_SUPERIORNAME	The name of the current user's supervisor.

Additional variables:

Variable	Effect
ELO_FLOWID	The ID of the current workflow.
ELO_FLOWNODE	An ID that is composed of the ID of the current workflow and the ID of the current node. For example 118.1.
ELO_NODEID	The ID of the current node.
ELO_TEMPLATE	The name of the form used.
ELO_OBJID	The ID of the ELO object used.
ELO_TICKET	The ELOas ticket.

You can also use values from the metadata form for completing form fields. The following variables can be used in addition to the variables for fields in the metadata and map fields:

Variable	Effect
IX_ID	Another option for reading the object ID.
IX_LOCKED	The user who currently has locked the entry.
IX_CREATEDATE	The filing date/creation date (including time) of the selected entry in ISO format. For example 20140827151800.
IX_MASKNO	The number of the metadata form used.
IX_MASKNAME	The name of the metadata form.

Keyword list

You can assign a keyword list to an input field or combo field via the *Keyword list* field. Date fields have the fixed keyword list *#DATE#*.

The screenshot shows a dialog box titled "Properties of the selected cell". It contains several fields:

- Field type:** Combo box
- Text:** (empty text box)
- Variable name:** IX_GRP_DEPARTMENT
- Keyword list:** A dropdown menu with "Keyword" selected. This dropdown is enclosed in a red rectangular box.
- Group name:** DEPARTMENT
- URL:** (empty text box)

There are several types of keyword lists for input fields and combo boxes:

Keyword: The keyword list of a field in the metadata can be used here. You can also enter keyword lists that do not refer to fields (*Global* keyword list, *Version number* keyword list, *Version comment* keyword list, and *Workflow* keyword list).

Dynamic keyword map: If you have selected the type *Dynamic keyword map*, you can enter data in the field using a dynamic keyword list.

- **Script name:** Enter the name of the Indexserver script that contains the dynamic keyword list in the *Script name* field.
-

Filter: Enter the variables for the required fields in the metadata and map fields in the *Filter* field. If you require multiple fields, you must separate them using commas.

Please note

You cannot use dynamic keyword lists in combo box fields.

ELOAS: Allows you to store a keyword list via an ELOas ruleset.

ELO Usernames: Allows you to select users and/or groups as a keyword list.

Proceed as follows to select a keyword list:

1. Select the keyword list type via the drop-down menu of the *Keyword list* field.

The corresponding input fields appear depending on the selected keyword list type.

2. Make the required settings.

Properties of the selected cell

Field type	Input
Text	<input type="text"/>
Variable name	<input type="text" value="IX_GRP_ORD_PROC"/> ▼
Keyword list	<input type="text" value="ELO Usernames"/> ▼
Options	<input checked="" type="radio"/> Users <input type="radio"/> Groups <input type="radio"/> Users and groups <input checked="" type="checkbox"/> Only visible <input type="radio"/> From every group <input checked="" type="radio"/> From group: <input type="text" value="GRP_ACCT"/> <input checked="" type="checkbox"/> Autofill <input type="checkbox"/> Only list values allowed

You can also enable the following options for keyword lists in input fields:

Autofill: A pen icon appears in input fields when the autofill option is enabled. When completing the form field, suggestions that match the user's entry automatically appear from the stored keyword list. If this option is not enabled, a button appears next to the input field that expands to a drop-down menu for the keyword list.

Only list values allowed: When this option is enabled, only entries from the keyword list are allowed in the field.

Existing entries: If the *Existing entries* option is enabled for an input field, ELO suggests terms that have already been entered in this field.

The image shows a form with two input fields. The first field is labeled 'Nr.' and is empty. The second field is labeled 'City' and contains a dropdown menu. The dropdown menu is open and shows three suggestions: 'Philadelphia', 'New York', and 'Chicago'. The 'City' field and its dropdown are highlighted with a red rectangular border.

You can open the keyword list with the F7 key. The keyword list also opens automatically as soon as you enter data in a field. ELO then suggests terms that are related to your entry.

URL

You can enter the document GUID (including brackets) or the website URL in the *URL* field.

Information

This function is not available for folders.

The image shows a dialog box titled 'Properties of the selected cell'. It has several fields: 'Field type' is set to 'Link'; 'Text' contains 'ELO'; 'Variable name' is empty; 'Keyword list' is a dropdown menu; 'URL' contains 'https://www.elo.com/en-de.html'; and 'View type' is a dropdown menu.

A left-click on a link to a document opens the external standard browser and downloads the document.

Information

If the URL is left empty in the form designer, the form provides a link to the current document.

Right-click the link to open a context menu. You have the following options in the context menu:

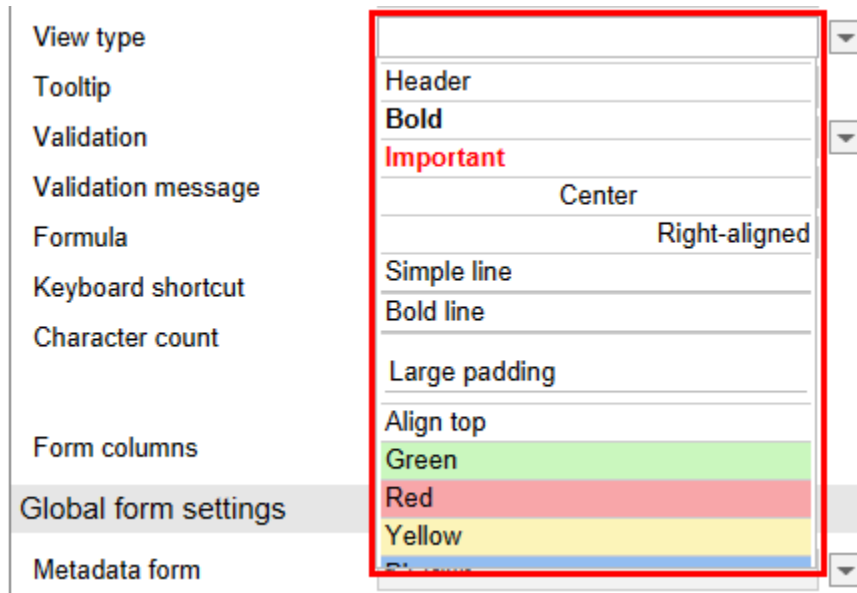
- Open link: Opens websites in the ELO browser. This does not work for documents.
- Open link in new window: Opens websites in the ELO browser. This does not work for documents.
-

Copy link to clipboard: Copies the link to the document or website to the Windows clipboard from where it can be moved to other locations.

Display mode

Use the drop-down menu of the *Display* field to change the layout of the selected cell.

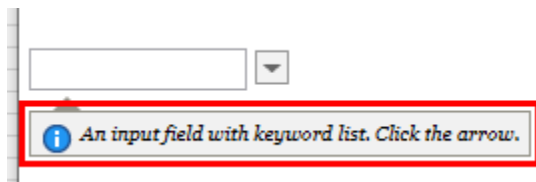
In addition, you can define the elements of the form using CSS properties. Use the *Form header scripts* component for this.



You can define your own classes that are available via the drop-down menu of the *Display* field.

Refer to the section Custom styles for more information.

Tooltip



You can enter a tooltip for all field types except for text fields. The tooltip is shown as soon as the user hovers over the respective field with the mouse. This way, you can provide the user with additional information and notes on the respective field.

Validation

URL	<input type="text"/>
View type	<input type="text"/> ▼
Tooltip	Input field with "notempty" validation
Validation	notempty ▼
Validation message	You must complete this field.
Formula	<input type="text"/>
Keyboard shortcut	b
Character count	20 <input type="text"/>

The *Validation* field is used to check the entry in the respective field. You can use validation to ensure that only numbers are valid as an entry or that the field cannot be empty, for example.

In the example, the value `notempty` is entered. This value triggers a check as to whether the field is empty. If the field is empty, a corresponding message is shown.

Validation message

Enter additional information for the user in the *Validation message* field. Enter the criteria that are required for the field to be completed correctly and for the validation to work.

In this example, the value `You must complete this field` is used.

Item number

✕

✕ The field is mandatory.
You must complete this field.

The user sees the text as a validation tooltip when text is entered in the field.

In this example, the text is "You must complete this field".

Formula

The *Formula* field is used for performing calculations in a field.

These are the valid arithmetic operators:

- Addition: +
- Subtraction: -
- Multiplication: *
- Division: /

Information

The typical calculation rules apply. Parentheses may be used.

The variables of the respective fields serve as wildcards for the actual values.

Example

The net value of an amount should be calculated in a field. We have added two input fields for the calculation:

- The *Gross value* field with the variable `IX_MAP_GROSS1`
- The *VAT rate (in %)* field with the variable `IX_MAP_VAT1`

The formula is generally: gross amount/(1+VAT rate)

In this case, the VAT rate must be calculated in decimals. For a tax rate of 19 %, you need to use a value of 0.19. The conversion to a decimal value is included in the formula so that the user can enter the VAT rate directly in percent.

Validation	<input type="text"/>	<input type="button" value="v"/>
Validation message	<input type="text"/>	
Formula	<input type="text" value="IX_MAP_BRUTTO1 / (1 + IX_MAP_UST1 / 100)"/>	
Keyboard shortcut	<input type="checkbox"/>	
Character count	<input type="text" value="20"/>	<input type="checkbox"/>
	<input checked="" type="checkbox"/> Read-only	
Form columns	<input type="text" value="1"/>	

The formula for the *Net amount* field is then:

```
IX_MAP_GROSS1 / (1 + IX_MAP_VAT1 / 100)
```

On-the-fly formulas

You can execute simple calculations in numeric fields. These fields must be configured with one of the validation parameters *num* or *amount*.

The formula must start with an equal sign (=). This is followed by the actual mathematical formula, e.g. "= 0.21 * 1234 + 57".

The calculation is done when the user presses the ENTER key or leaves the field.

These inline formulas are limited to numerical computations. Other variables are not available.

ESum

For fields that are created using JS_ADDLINE, use the *ESum* function to add up all fields with a common variable name.

You must use the following format:

```
ESum(IX_MAP_<Name>)
```

or:

```
ESum(WF_MAP_<Name>)
```

Example

View type	<input type="text"/>	▼
Tooltip	<input type="text"/>	
Validation	amount nk:2	▼
Validation message	<input type="text"/>	
Formula	ESum(IX_MAP_ZWSUM)	
Keyboard shortcut	<input type="text"/>	
Character count	10	<input type="text"/>
	<input checked="" type="checkbox"/>	Read-only

In this example, there is a JS_ADDLINE line with the *Subtotal* field that is linked to the *IX_MAP_STOT1* variable. The formula *ESum(IX_MAP_STOT)* adds up all values from the fields (1 to n) that are generated with the variable *IX_MAP_STOT*.

Keyboard shortcut

Use the *Keyboard shortcut* field to assign a shortcut that lets you jump directly to the respective field.

Validation message	You must complete this field.	
Formula	<input type="text"/>	
Keyboard shortcut	b	
Character count	20	<input type="text"/>
	<input type="checkbox"/>	Read-only
Form columns	1	

To do so, enter the letter you want to assign to the respective form field to the *Keyboard shortcut* field. In the form, jump to the respective form field with the keyboard shortcut ALT+<KEY>.

Information

The keyboard shortcut only works when the form is used directly in the browser. Do not use any shortcut that is already used by the respective browser.

Character count

The two fields next to *Character count* perform different tasks depending on the field type.

You can use these fields to enter the size of input fields and editor fields.

For more information, refer to the *Toolbar* section.

Read-only

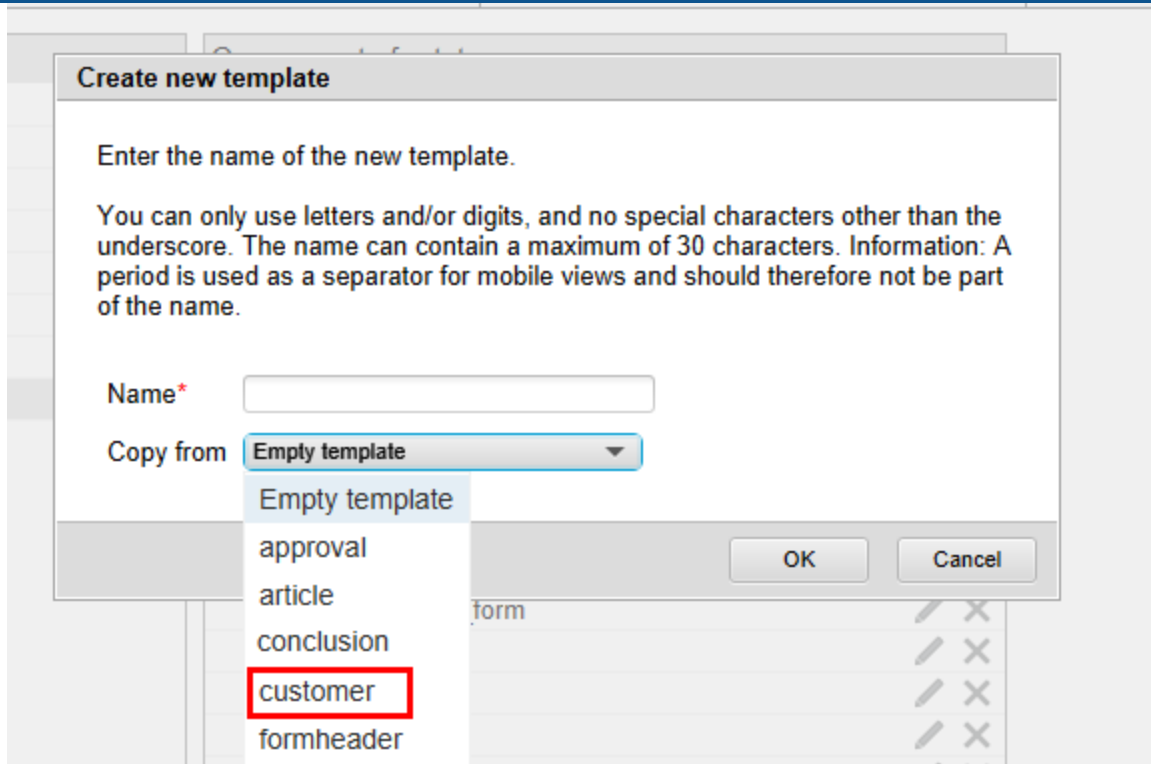
All fields that the user can enter data into can be set as read-only. Select the *Read-only* check box to write-protect a field.

Example

You want the first user (node A) to be able to complete the *customer* template. Another user (node B) should be able to see the template but not change it.

The screenshot shows the 'Form designer' window. The top toolbar includes options like 'New line', 'Delete line', 'Input', 'Check box', 'Image', 'JSAddLine', 'Delete', 'New column', 'Delete column', 'Date', 'Radio button', 'Signature', 'JSRemoveLine', 'Cut', 'Merge', 'Split', 'Text', 'Combo box', 'Link', 'Copy', 'Button', 'Relation', and 'Paste'. The main area displays a form with a table containing 'Employee data' with fields for 'Employee', 'Department', 'Supervisor', and 'Order date'. The 'Properties of the selected cell' panel on the right shows settings for 'Field type' (Text), 'Variable name', 'Keyword list', 'URL', 'View type', 'Tooltip', 'Validation', 'Validation message', 'Formula', 'Keyboard shortcut', 'Character count', and a 'Read-only' checkbox.

1. First, create the template *customer*.



- Now copy the *customer* template. To do so, select the corresponding template under *Copy from* in the *Create new template* dialog box.

Please note

Subsequent changes to the original template must also be made in the copy.

Components for material_order		
	approval	
	approval_prot	
	article	
	article_prot	
	conclusion	
	customer	
	customer_prot	
	formheader	
	orderedby	
	orderedby_prot	

- Name the copied template *customer_prot* to ensure that the two templates can be clearly distinguished from one another.

Validation message	<input type="text"/>
Formula	<input type="text"/>
Keyboard shortcut	<input type="checkbox"/>
Character count	<input type="text" value="20"/> <input type="checkbox"/>
	<input checked="" type="checkbox"/> Read-only
Form columns	<input type="text" value="1"/>

Global form settings

- Select the *Read-only* setting for all fields of the *customer_prot* template.

Employee data **customer**

Employee Department

Supervisor Order date

Employee data **customer_prot**

Employee Department

Supervisor Order date

- Finally, assign the *customer* template to node A and assign *customer_prot* to node B in the workflow template.

Form columns

The *Form columns* field shows you how many columns the respective selected cell spans. This is relevant, for example, when the respective cell is connected to multiple cells.

Global form settings

The following setting options are located in the *Global form settings* area.

Metadata form

Use the *Metadata form* field to define which metadata form should be used to store the form data.

Information

Newly created metadata forms do not appear immediately in the form designer. If necessary, run the *Refresh* function in the form designer.

Alternative: Restart the *ELO Indexserver* and then *ELO Web Forms Services*. Select the corresponding metadata form via the drop-down menu of the field.

Name

The name of the template is listed in the *Name* field. Change the name here if required.

Map name (tables only)

Global form settings

Metadata form	8: Order	▼
Template name	article_table	
Map name	ARTICLE	
Languages		▼
Translation variable (prefix)		
	<input type="checkbox"/> Limited variable access	

Use the *Map name* field to specify the name that the table data will be saved as. The data is stored via special map fields. One map field is created per table line. Each of the map fields contains the name entered in *Map name* as well as a sequential number. The contents of the map fields are displayed in the metadata under *Additional information*.

Please note

The map name must not contain special characters or spaces.

Languages

The *Languages* field is used to determine which languages you want the control elements of the form to be available in. Select the languages via the drop-down menu of the field.

The image shows two parts of a user interface. The top part is a form titled "Global form settings" with the following fields:

- Metadata form: 11: Project file
- Template name: basic
- Languages: de.en.fr (highlighted with a red box)
- Translation variable (prefix):
- Limited variable access:
- Realign columns:

 A red arrow points from the "de.en.fr" field to a dropdown menu below. The dropdown menu is currently set to "English" and shows a list of options: Deutsch, English, and Français. The "English" option in the list is also highlighted with a red box.

When completing the form, the user can select the language to be used from a drop-down menu.

In this example, the value `de.en.fr` in the *Languages* field results in the languages German, English, and French being offered in the form.

Information

The image shows a file explorer view of the "Configuration" folder under "ELOWf Base". A red box highlights the "Configuration" folder. A red arrow points from this folder to a "Metadata" window. The "Metadata" window has tabs for "Basic", "Extra text", and "Options". The "Extra text" tab is active, showing the following metadata entries:


```
#Thu Jul 29 07:21:14 CEST 2021
JavaClientDateFormat=true
mainlanguage=en
encodeAutoComment=true
version=21.00.001
showScriptErrors=true
```

 The entry `mainlanguage=en` is highlighted with a red box.

The default language is defined by the entry *mainlanguage* in the extra text of the metadata in the *Configuration* folder under *ELOWf Base*.

This is the reference language used when searching for translations.

The following options can be used to translate the interface texts:

- Translation table (for more information, refer to the chapter [Translation](#) under *Translation table* in the documentation *ELO Java Client administration*.)
- Properties files (see following section)

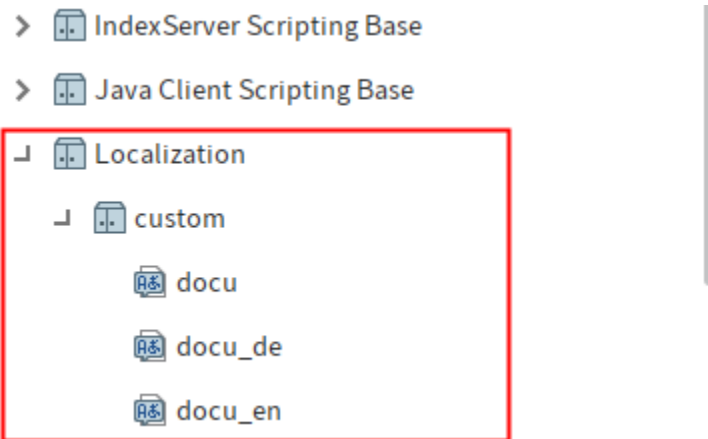
Information

ELOWf caches translations for performance reasons. If you modify the translation tables or properties files, reload the ELOWf module for the changes to take effect.

When modifying the properties files, reload the ELO Indexserver as well.

Translation variable (prefix)

If you want to provide translated texts using scripts, use the *Translation variable (prefix)* field. Simply save the text files to ELO with the *.properties* file extension.



The following applies for properties files:

- Character encoding: UTF-8
- Path in ELO: Administration//Localization//custom OR system
- One for each language: A properties file with the corresponding country code (de, en, fr, etc.)

The contents of a properties file consist of a list with key-value pairs according to the following convention:

```
<Prefix>.<Key 1>=<Value 1>
<Prefix>.<Key 2>=<Value 2>
<Prefix>.<Key n>=<Value n>
```


File Edit Format View Help

```
pf.messageDays=Please enter the desired number of days:  
pf.headDays=number of days|
```

Example:

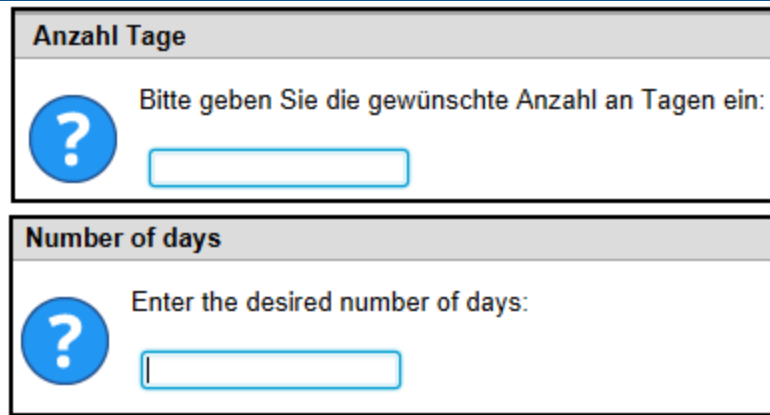
```
pf.messageDays=Enter the desired number of days:  
pf.headDays=Number of days
```

In the form itself, the prefix must be entered in the *Translation keys* field that corresponds to the respective entries in the properties file. You can then use scripts to call entries with the same prefix.

```
23  
24     function JS_DAYS () {  
25         var params = {title: elo.locale.store['pf.headDays'],  
26                       message: elo.locale.store['pf.messageDays'],  
27                       type: "prompt",  
28                       width: "300",  
29                       onOk: JS_UPDATE};  
30         $msg (params);  
31     }  
32  
33     function JS_UPDATE (text) {  
34         $update ("IX_GRP_DAYS",text);  
35     }  
36 </script>  
37
```

Please note

You need to restart the *ELOix* and *ELOWf* modules so that ELOWf can load the properties file.



Anzahl Tage

Bitte geben Sie die gewünschte Anzahl an Tagen ein:

Number of days

Enter the desired number of days:

The texts in the properties files are read via scripts depending on the language selected.

One way is to use the method `api.helpers.Text.getText`.

Example:

```
eloAlert(api.helpers.Text.getText('bar.mynotification'))
```

Limited variable access

If the *Limited variable access* option is enabled, only the contents of the fields in the metadata that are required for displaying the form are transferred when processing the form data.

If the *Limited variable access* option is disabled, the contents of fields in the metadata that are ignored by the form can be read out, such as via the browser source text of the form. For example via the browser source text of the form.

Information

For dynamic templates (e.g. when using the `JS_ADDLINE` variable) and in table templates, the *Limited variable access* option is not permitted.

Realign columns

The *Realign columns* option must be selected in table templates for them to work.

Integrate a form into a workflow

To use forms in workflows, you need to specify in the workflow template which form templates are to be used at which nodes. This is a distinct advantage of splitting workflow forms into templates. This makes it possible to specify which workflow participants will be shown which parts of the form.

Templates can be integrated via user nodes and the start node. The templates integrated via user nodes are shown to the workflow participant who is currently processing the workflow.

The owner (user who started the workflow) of a workflow can show templates integrated via the start node.

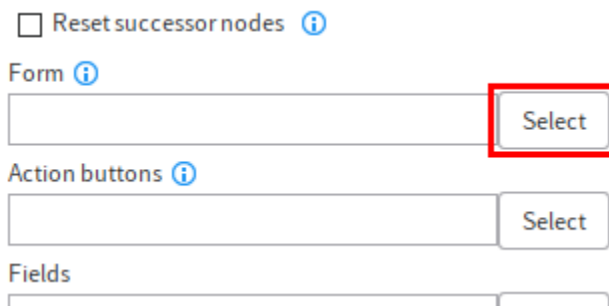
Example

The following example describes how to integrate templates via a person node. For the example, the *Material order* form and the corresponding *Material order* workflow template and *Material order* metadata form were created.

1. Open the desired workflow template.
2. Select *Edit workflow template* to switch to edit mode.
3. Select the desired user node.

The settings of the selected node can now be edited.

4. Navigate to *Additional options* and then to the *Form* field.



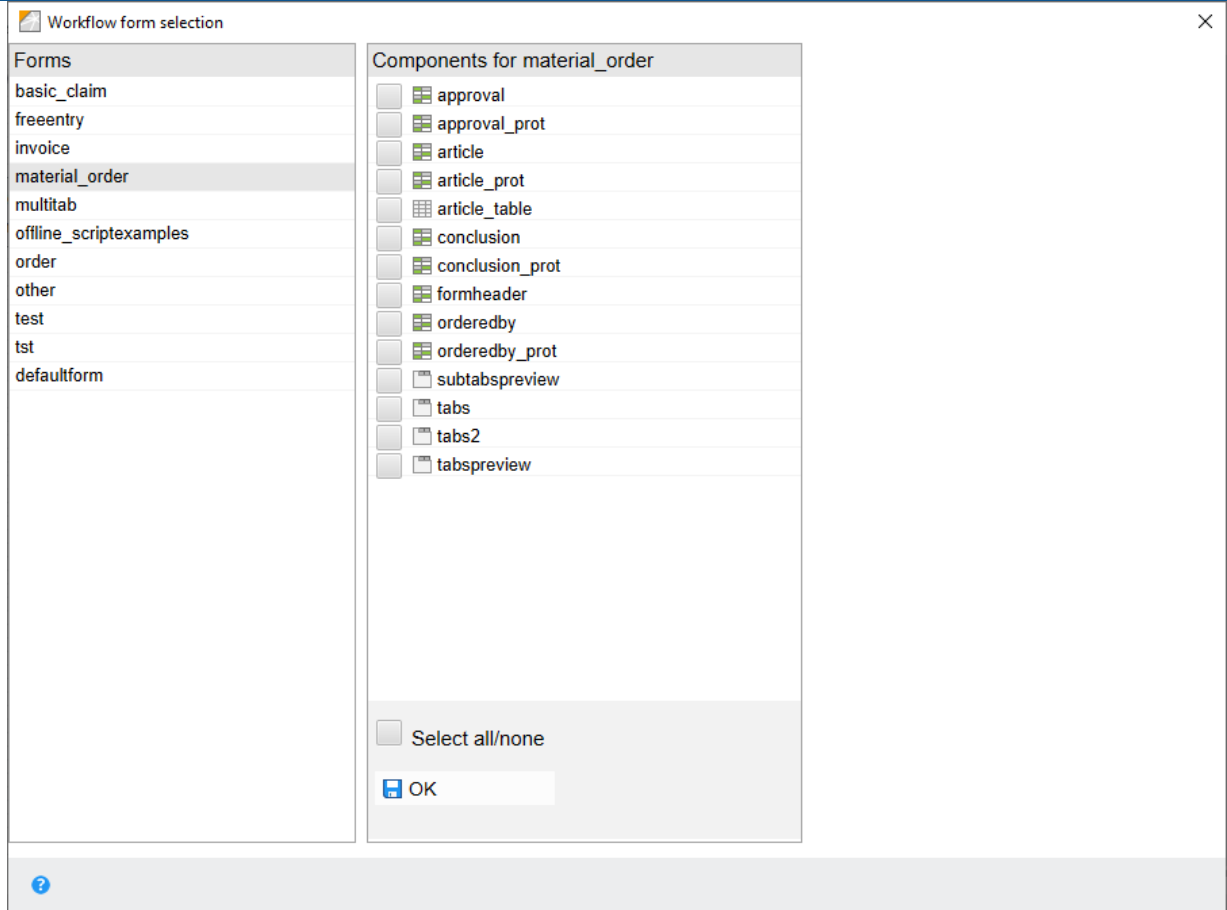
The screenshot shows a configuration interface with the following elements:

- A checkbox labeled "Resetsuccessornodes" with an information icon.
- A section titled "Form" with an information icon, containing a text input field and a "Select" button. The "Select" button is highlighted with a red rectangular border.
- A section titled "Action buttons" with an information icon, containing a text input field and a "Select" button.
- A section titled "Fields" with a text input field.

5. Select the *Select* button (to the right of the *Form* field).

The *Workflow form selection* dialog box opens. The available forms appear in the *Forms* column.

6. Select the form you want to use.



The *Components for* column with the available templates appears in the right area of the dialog box.

7. Select a template.

Information

You can select multiple templates for one node.



The selected template is moved up in the *Components for* column. The check box in front of the respective template is now enabled. There are two black arrows to the right of the template.

8.

Select *OK*.

Reset successor nodes ⓘ

Form ⓘ

[material_order(approval)]

Action buttons ⓘ

Fields

The *Workflow form selection* dialog box closes. The template selected above is now integrated into the respective workflow node.

- Confirm the changes to the workflow template in the *Workflow designer* dialog box with *OK* or *Apply*.

From now on when you start a workflow with the corresponding template, the selected template is shown to the responsible workflow participant.

Change order

Components for material_order		
<input checked="" type="checkbox"/>	approval	↑ ↓
<input checked="" type="checkbox"/>	formheader	↑ ↓
<input checked="" type="checkbox"/>	orderedby_prot	↑ ↓
<input checked="" type="checkbox"/>	article_prot	↑ ↓
<input type="checkbox"/>	approval_prot	
<input type="checkbox"/>	article	

Optional: Change the order of the templates via the arrows to the right of the selected templates. The order set in the *Workflow form selection* dialog box (from top to bottom) corresponds to the order displayed in the form.

Integrate a form without the dialog box

You can enter forms or their respective form templates directly into the *Form* field.

You must use the following pattern:

```
[formname(templatename1,templatename2,...)]
```

Please note

All names must be lowercase.

The order (from left to right) corresponds to the order displayed in the form.

In the example, the expression is entered as follows:

```
[materialorder(formheader,approval, customer_prot,items_prot)]
```

Integrate form (gen. 2)

Starting with ELO 21, you can create second generation forms. This is done via package administration in the ELO Administration Console.

You will find more information on creating forms under [ELO packages > Metadata > Forms](#).

To integrate the forms (gen. 2) into a workflow template, the following requirements must be met:

- There is at least one package
- This package has at least one metadata form
- A view/form (gen. 2) has been created for this metadata form
- There is a workflow template

To integrate a form (gen. 2) into a workflow template, you can proceed in the same way as described in [Integrate a form without the dialog box](#). Follow the steps below.

1. Open the desired workflow template.
2. Select *Edit workflow template* to switch to edit mode.
3. Select the desired user node.

The settings of the selected node can now be edited.

4. Navigate to *Additional options* and then to the *Form* field.

Reset successor nodes ⓘ

Form ⓘ

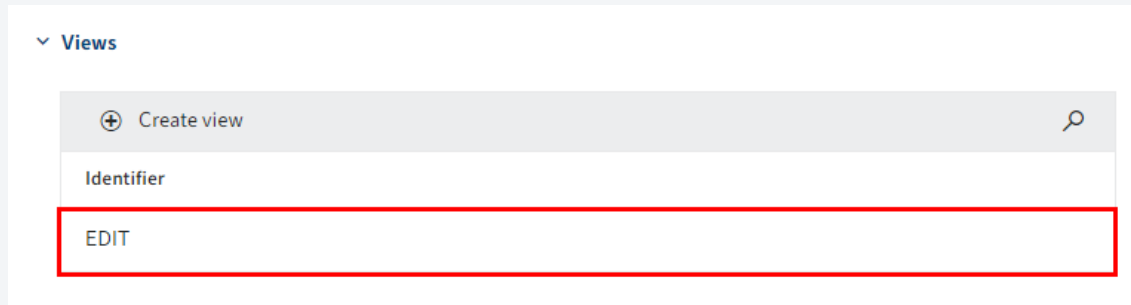
edit

Action buttons ⓘ

Fields

5. Enter the name of the form (gen. 2) to be used for the metadata form in lowercase letters.

Information



You will find the name of the form in the configuration of the respective metadata form (*ELO Administration Console > Package administration > Package > Metadata form*) under *Views*.

6. Confirm the changes to the workflow template in the *Workflow designer* dialog box with *OK* or *Apply*.

As soon as you start a workflow with the corresponding template, the selected form is shown to the responsible workflow participant.

Please note

The entry selected to start the workflow must have been filed with the corresponding metadata form. Otherwise, it will not be possible to connect to the correct form.

If you select an entry with a metadata form, and a form (gen. 2) with the same name exists, this form is shown.

Save form data

You can evaluate and process the data in an ELO form in various ways. For example, you can use the data to determine what the next step is in the current workflow. In addition, the next editor in the workflow can view the data so that the responsible user knows how to proceed with the workflow. Form data in ELO can also be analyzed and used in other business processes.

The data needs to be saved so that it can be processed later. You can do this in ELO with one of the following options:

- Via the metadata
- Via IX map fields
- Via WO map fields or BLOB fields
- Via WF map fields

Via the metadata

Field type	Input
Text	<input type="text"/>
Variable name	IX_GRP_INPUT
Keyword list	Short name
URL	Extra text
View type	Document date
Tooltip	Note on the current node
Validation	Personal identifier
Validation message	End of deletion period
Formula	City
Keyboard shortcut	ClaimNo
Character count	Damage
	Event date
	FirstName
	InsNo
Form columns	1

To save the data entered in the form field using metadata, the form field must be connected to a field. To do this, you need to select the *Variable name* field from the drop-down menu.

Information

The number of fields in the metadata is limited to 200. You need to create the fields you require in advance. Fields cannot be created dynamically.

Map fields

Map fields provide an additional option to save data. Map fields are especially necessary if you want to use dynamically generated forms (rows created with JS_ADDDLINE).

IX map fields

Properties of the selected cell

Field type	Input
Text	<input type="text"/>
Variable name	<input type="text" value="IX_GRP_ARTICLE1"/> ▼
Keyword list	<input type="text" value=""/> ▼
URL	<input type="text"/>
View type	<input type="text"/> ▼

IX map fields are processed by the ELO Indexserver. The contents of IX map fields are saved to the *map_objects* database. IX map fields are linked to the respective entry. IX map fields are an extension of the metadata. Users with the relevant permission (*Show "Additional information"*) can view the contents of the IX map fields in the *Metadata* dialog box via the *Additional information* tab.

To use an IX map field, enter a corresponding call in the *Variable name* field in the form designer. Use the following syntax:

```
IX_MAP_<NAME><STARTVALUE>
```

The name must not contain special characters or spaces. You need to enter a start value if you want to create a dynamic form. ELO automatically increments the value if the line containing the IX map field is duplicated using JS_ADDDLINE.

The following also applies:

- <NAME>: The last character in the name cannot be 0.
- <STARTVALUE>: The start value cannot be 0, i.e. it must be greater than zero. The value must therefore be greater than zero.

Example

Properties of the selected cell

Field type: Input

Text:

Variable name: IX_MAP_ITEM1

Keyword list:

Metadata

Basic | Extra text | Options | Permissions | Version history | Additional information

Current form: Order

Name	Value
ITEM1	A4 folder, black
ITEM2	Labels 60 mm, 100x
ITEM3	Subject dividers, 100x

As the line with the IX map fields was duplicated using a JS_ADDLINE button, there are multiple entries (ARTICLE1 and ARTICLE2 etc.) for every IX map field in the *Metadata* dialog box. The number at the end of the name is incremented automatically in ELO.

Information

If you are storing a lot of values in IX map fields, this will increase the time needed to access the documents.

WO map fields

Properties of the selected cell

Field type: Signature

Text: Sign

Variable name: IX_BLOB_WO_TEST

Keyword list:

URL:

View type:

WO map fields are designed for signature fields/BLOB fields (Binary Large Objects). WO map fields are processed by the ELO Indexserver. The contents of WO map fields are saved to the

map_formdata database. WO map fields are linked to the respective entry and cannot be overwritten (WO = write once).

To use a WO map field, enter a corresponding call in the *Variable name* field in the form designer. Use the following syntax:

```
IX_BLOB_WO_<NAME>
```

The name must not contain special characters or spaces.

WF map fields

Properties of the selected cell

Field type	Input
Text	<input type="text"/>
Variable name	<input type="text" value="WF_MAP_STATUS1"/> ▼
Keyword list	<input type="text"/>
URL	<input type="text"/>
View type	<input type="text"/> ▼

WF map fields work in a similar way to IX map fields. However, when using WF map fields, the form data is saved in the workflow object and not in the metadata. This means the data is only related to the current workflow and cannot be called in repeated runs of the same workflow template.

Information

WF map fields cannot be read via the *Metadata* dialog box. Outside of the form, WF map field data can only be accessed via the ELO Indexserver API.

Validation

Validation is the function that checks data to ensure it complies with the input rules defined in advance. To create these input rules, number fields and text fields can be used as input fields.

Tooltip	<input type="text"/>
Validation	<input type="text"/> ▼
Validation message	{no default value}
Formula	Date
Keyboard shortcut	Text, min. 3 characters, max. 10 characters
Character count	Not empty
	Not empty when passed forward
	Numeric
Form columns	Numeric, integer
Global form settings	Numeric, 2 decimal places
Metadata form	Numeric, minimum value 100, maximum value 500
Template name	Check numeric input
Languages	Amount

The drop-down menu in the *Validation* field contains a set of predefined validation rules. Select the desired rule.

Alternatively, you can configure your own validation rules. The different validation parameters are explained below:

date

The validation parameter *date* is intended for date fields. Fields with this validation rule only accept date and time entries. If the date entered is incorrect, a tooltip error message will appear.

Date format

If you use the ELO Java Client, forms will use the same format. This format has either a specific pattern or depends on the respective country setting.

Date fields accept the following date formats:

- YYYYMMDD
- YYYY-MM-DD
- YYYY.MM.DD HH:mm
- DD.MM.YYYY
-

DD.MM.YYYY HH:mm

- DD.MM.YY

(Y = year/M = month/D = day/H = hour/m = minute)

Information

Seconds are not saved.

Please note

If you are not using the ELO Java Client, you cannot use other date/time formats.

When writing, inserting to or updating the date field, it is recommended to use the same format. However, it is also possible to use:

- The alternative format specified in the ELO Java Client
- The universal ISO 8601 format (e.g. "2014-09-08T08:02:17")

Internationalization and time zones

If you log on with a client in another time zone, the `IX_DOCDATE` and `IX_CREATEDATE` will be adapted to the local time.

For example, a document with a creation date like "Nov 12, 2021 7:08" will be changed to "Nov 12, 2021 10:08 am" if the user logs on in a time zone three hours later. This applies as well if it was a date "only", which will be considered to be the date at midnight. For example, "Nov 12, 2021" is the same as "Nov 12, 2021, midnight".

Information

All other dates in the form will be left unchanged.

Years

When using the double-digit format (DD.MM.YY), all values greater than or equal to 70 are treated as 19YY and all other values are treated as 20YY. Example: 70 becomes 1970, and 11 becomes 2011.

text

The validation parameter *text* is intended for text fields. It only makes sense to use this parameter in combination with *min* and/or *max*.

num

The validation parameter *num* is designed for number fields. This parameter can be used in combination with *min* and *max*.

nk

Available for the validation parameters *num* and *amount*. Specifies the number of decimal places for numeric inputs. Data entered in this field is automatically changed to this number when you leave the field.

Example 1: `num nk:0` = only whole numbers without decimal places

Example 2: `amount nk:2` = numbers with two decimal places

Examples

The following lists illustrate the different behavior of *num* and *amount*:

`num nk:2` behaves as follows (in English locale):

- "12346" is displayed as "12346.00"
 - No thousands separator
- "12,346" is displayed as "12.35"
 - Interprets both the comma and the period as a decimal separator
- "12.346" is displayed as "12.35"

`amount nk:2` behaves as follows (in English locale):

- Displays a thousands separator
- "12346" is displayed as "12,346.00"
 - Interprets the period as a thousands separator
- "12,346" is displayed as "12,346.00"
- "12.346" is displayed as "12.35"

Amount fields can accept input with or without thousands separators but will always display it with them.

Customize with a script

The following flag can be set in the header script:

```
ELO.Configuration.Amount.noThousandSep = true;
```

If the flag is set, the amount fields will behave similarly to num fields and interpret the last separator as a decimal separator, no matter whether it is a period or a comma. This is especially important if the data supplied by the server (e. g. map fields) does not use uniform separators, but

instead contains numbers with both periods and commas in the database due to different import processes.

Format

What is used as thousands/decimal separators?

If the ELO Java Client is being used, the format can be set via `> Number formats`.

There are two options available:

- Standard: (recommended, is also language dependent).
- User-defined: You can also specify custom separators (not recommended, as the separator stays the same even if you switch to another language).
- Otherwise, the default locale is used.

If you are unsure of which separator is used, you can verify the setting in the form header. These are stored in the following variables:

```
ELO.Configuration.Amount.ThousandSep = ...  
ELO.Configuration.Amount.DecimalSep = ...
```

min

The syntax for the parameter *min* must be as follows:

```
min:<numericvalue>
```

The parameter *min* has different properties depending on the field or combination with other parameters. You have the following options:

- Combination with a text field: Determines the minimum character length. If you enter a value above 0, the field becomes mandatory.
- Combination with a numeric field or amount field: Determines the lowest possible numeric value.

Example

```
text min:1
```

max

The syntax for the parameter *max* must be as follows:

```
max:<;numericvalue>
```

The parameter *max* has different properties depending on the field or combination with other parameters. You have the following options:

-

Combination with a text field: Determines the maximum character length.

- Combination with a numeric field or amount field: Determines the highest possible numeric value.
- Combination with a button: For a JS_ADDLINE button, this value determines the maximum number of additional lines that the user is allowed to create.

Example

```
num max:10
```

amount

The validation parameter *amount* is intended for amount fields. Like numeric fields, amount fields expect a numeric input. However, amount fields try to interpret and express these numbers as amounts. The format of the amounts displayed depends on the language you have selected in the client.

Example

If the language of the client is set to German and you enter 5999,99, the amount will be displayed as follows:

```
5,999.99
```

If you save the amount and then switch the language of the client to English, the amount will be displayed as follows:

```
5.999.99
```

notempty

The validation parameter *notempty* makes the field mandatory without defining a minimum input value. A corresponding validation message will appear.

The form can only be saved or forwarded once all mandatory fields have been completed.

notemptyforward

If the validation parameter *notempty* is replaced with the parameter *forward*, you can save the form even if you have not entered a value in the mandatory field. You can only forward the form when all mandatory fields have been completed.

asname

Available for buttons. If you want to call an ELOas function from a button, the name of the ruleset must be entered here. You can add optional parameters (param2, param3).

param2 and param3

Available for buttons. With these two expressions, parameters can be entered for the ELOas function call. If the value starts with an exclamation mark (e.g. !123), the value is transferred directly without the exclamation mark. Otherwise, the current value of the input field will be transferred along with its variable name (e.g. IX_GRP_RENUM).

copy

Available for all fields. This value can have the values *true* or *false*, and determines whether the current field contents are also copied when an input field is copied. This parameter cannot be combined with the parameter count.

count

Available for num. This setting can be used to automatically create line numbers for copied input fields. If the validation field contains the entry `count:auto`, the value in the new line will be that of its preceding cell plus one. This setting cannot be combined with the parameter copy.

lines

Available for buttons. For a button with the JS_ADDLINE function, this parameter specifies how many lines are copied. The default value is one line.

add<*>

Available for input fields. When selecting multiple terms from a keyword list, this parameter allows you to attach the value in the input field to the last entry instead of overwriting it.

When using this parameter, a space will be used to separate the entries. You can also define another character as a separator. An underscore will always be interpreted as a space.

For example, if you enter `add, _` a comma and a space are placed after the preceding entry.

Validation message

Enter your own text in the *Validation message* field in case the validation does not work. This text can be translated using the translation table or properties files.

Custom validation function

In addition to the standard validation functions, it is possible to incorporate custom validation functions using scripts. Validation functions must be named according to the following convention:

```
JS_VAL_<name of function> (fieldName, fieldValue, param)
```

For an invalid field, the function must return the error to be displayed as a non-empty string.

Properties of the selected cell	
Field type	Input
Text	<input type="text"/>
Variable name	IX_MAP_A <input type="button" value="v"/>
Keyword list	<input type="text"/>
URL	<input type="text"/>
View type	<input type="text"/> <input type="button" value="v"/>
Tooltip	<input type="text"/>
Validation	JS_VAL_DIV:3 <input type="button" value="v"/>
Validation message	<input type="text"/>
Formula	<input type="text"/>
Keyboard shortcut	<input type="text"/>
Character count	14 <input type="text"/>

In the example, the custom validation function JS_VAL_DIV is entered in the *Validation* field and a parameter with the value "3" is added with :3.

The following script is used in this example:

```
function JS_VAL_DIV(name, vlaue, param) {
  var num = +value
  if (num % param != 0) {
    return "Not divisible by " + param
  }
}
```

Edit form header scripts

```

1 <script type='text/javascript'>
2
3   function JS_VAL_DIV(name, value, param){
4     var num = +value;
5     if (num % param !=0 ){
6       return "Not divisible by " + param;
7     }
8   }
9 
```

Number

✖ Not divisible by 3

↔

Number

In the example, entries in the field are then checked to see whether they are divisible by the value "3". If validation fails, the corresponding message appears as a tooltip under the field.

This type of validation can be used in addition to other validation constraints. The only limitation of this functionality is that the scripts have to be synchronous.

Custom filter functions

Sometimes it is useful to convert manual entries into the corresponding format to ensure that the validation will work. You can do so by integrating your own filter functions in the *Validation* field. Filter functions must be named according to the following convention:

```
JS_FILTER_<Name of the function> (front, inserted, tail, param)
```

Here is example of how to automatically uppercase the input:

```
function JS_FILTER_Uppercase(front, inserted, tail) {  
  return inserted.toUpperCase();  
}
```

In this case, the return value is a string and modifies what was inserted. However, it is also possible to modify what comes before and after the entry, by returning an array with the new values ([front, inserted, tail]).

Can I turn validation on or off depending on the use case?

No. Validation always take place. It also makes sense. Even if you have a workflow with a "Cancel" operation, the form still has to be saved. And in order to be saved, it has to contain valid data.

Rather than trying to disable validation, we recommend using `nextClicked(...)` to simply enter dummy values where needed before the validation takes place.

Custom styles

You can enter additional styles in the `<style>` part of the header data of the form (*Edit form header scripts* component). The usual CSS rules apply.

Information

In the extra text of the *Classes* folder, you can also add these styles to the list of default styles. These styles are automatically available by means of the keyword list.

The following sections explain how to define your own style class and use it in a form.

Define style settings

```
54
55 </script>
56
57 <style type='text/css'>
58   .formblack {
59     color: white;
60     background-color: black;
61   }
62
63 </style>
```

The style settings are defined in the form designer. The styles must be entered in the corresponding header area (*Edit form header scripts* component) of the form. The class used in the example is named *formblack*. It is assigned the background color value *black*.

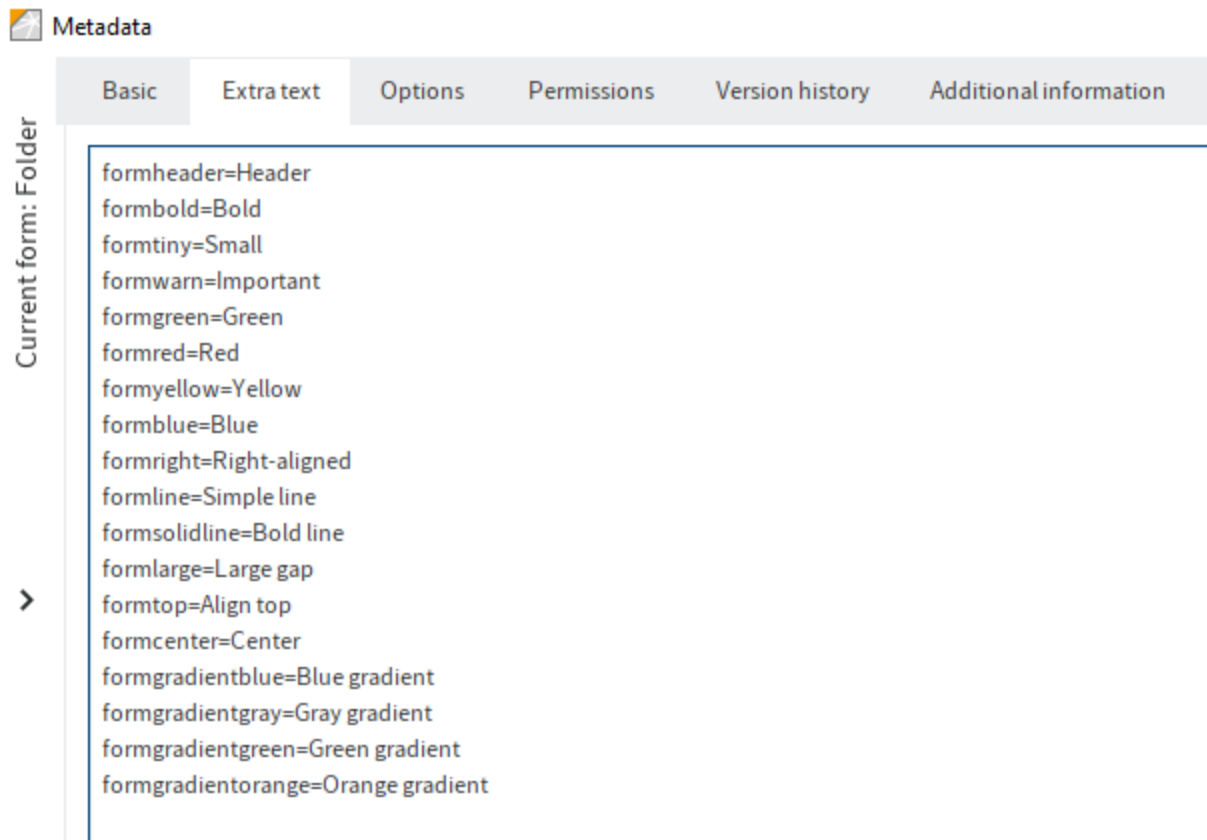
It would be possible to use the style in the form now. However, you would need to enter the style manually (in the respective *Form designer > Desired template > Properties of the selected cell > Display* form field).

In order to be able to select the style from the drop-down menu of the *Display* field, follow the steps below.

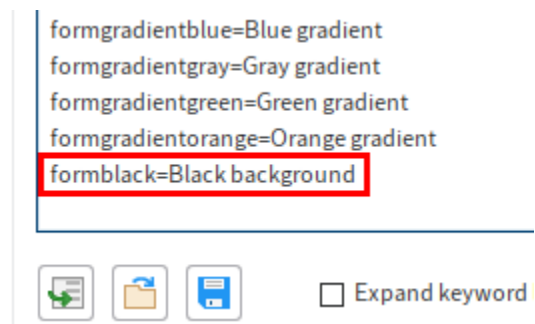
Configure class in ELO

There are several default display classes in ELO. An administrator is able to add additional classes.

1. Navigate to the folder *Classes* (*Administration//ELOwf Base*) in ELO.
2. Open the *Metadata* dialog box for the *Classes* folder.
3. Click the *Extra text* tab.



You see all display classes defined so far on the *Extra text* tab.

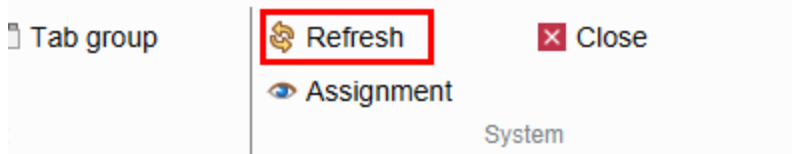


4. Enter the new class at the end of the list. Use the following syntax:

```
<class name>=<class display name>
```

5. Select OK to save changes and close the dialog box.

6. Open the form designer.



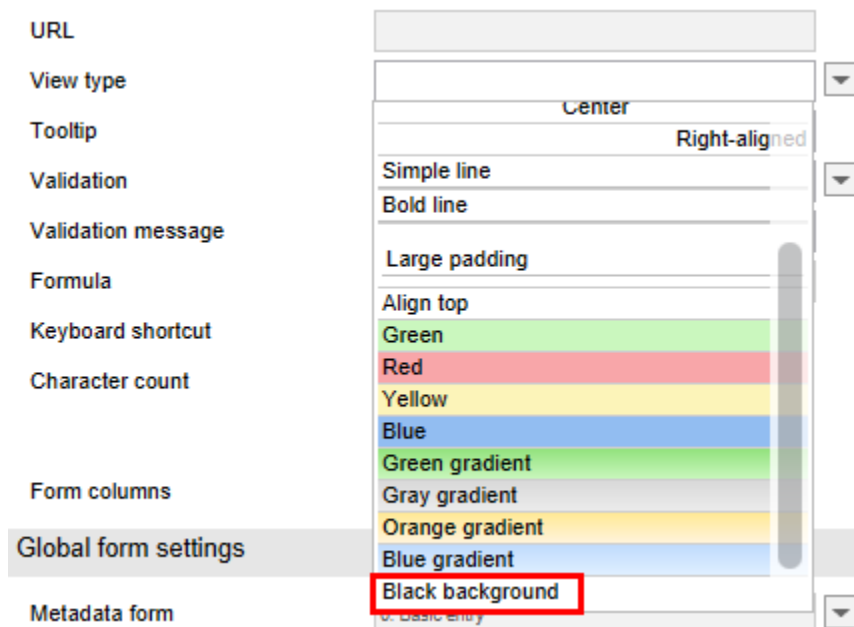
7. Select the *Refresh* button on the toolbar.

The new display class is now available.

Apply style

If you have defined a new style and created the corresponding class, you can use the style in the form.

1. Open the form designer.
2. Select the form you want to use the style in.
3. Open the template you want to use the style in.
4. Select the form field you want to assign the style to.
5. Open the drop-down menu of the *Display* field (*Properties of the selected cell*).

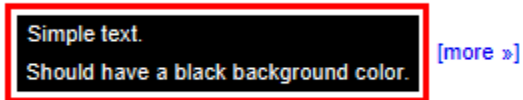


6. Select the new style from the list.

Information

You cannot preview added styles in the form designer. Use the *Save and preview* function to check whether the style you selected is displayed as required.

7. Select *Save*.

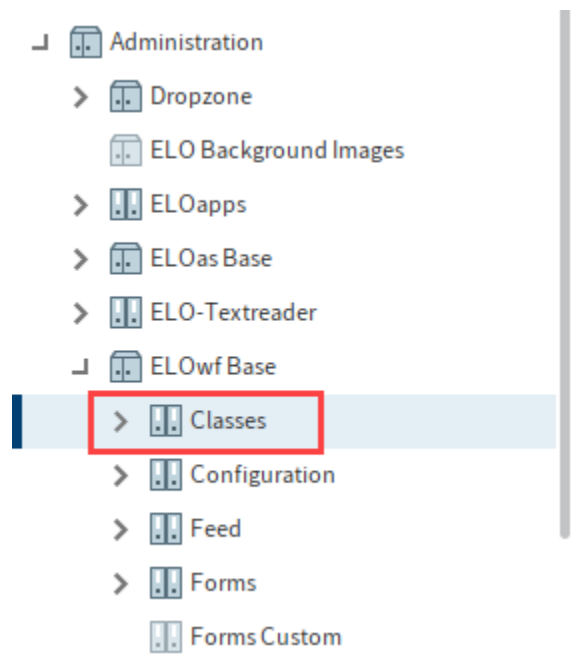


The new style is assigned to the selected field.

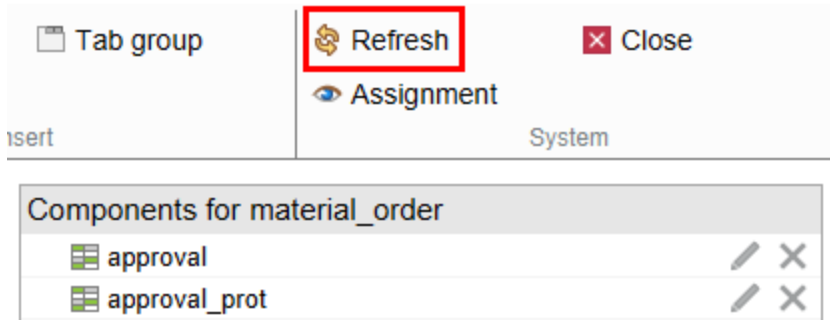
Integrate CSS files

If you want to use CSS styles in multiple forms, it makes sense to save the styles in CSS files. You can then manage the CSS files in ELO and load them to forms.

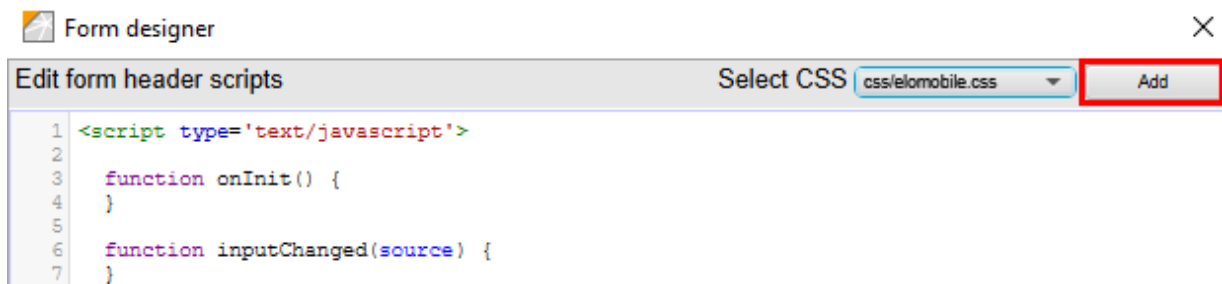
1. Save the CSS files to ELO. Use the following path:



Administration//ELOWf Base//Classes



2. Open the form designer.
3. To load the CSS files, select *Refresh* in the form designer.
4. Open a form.
5. Select *Edit form header scripts*.
6. Select a CSS file from the *Select CSS* drop-down menu.



7. Select *Add*.

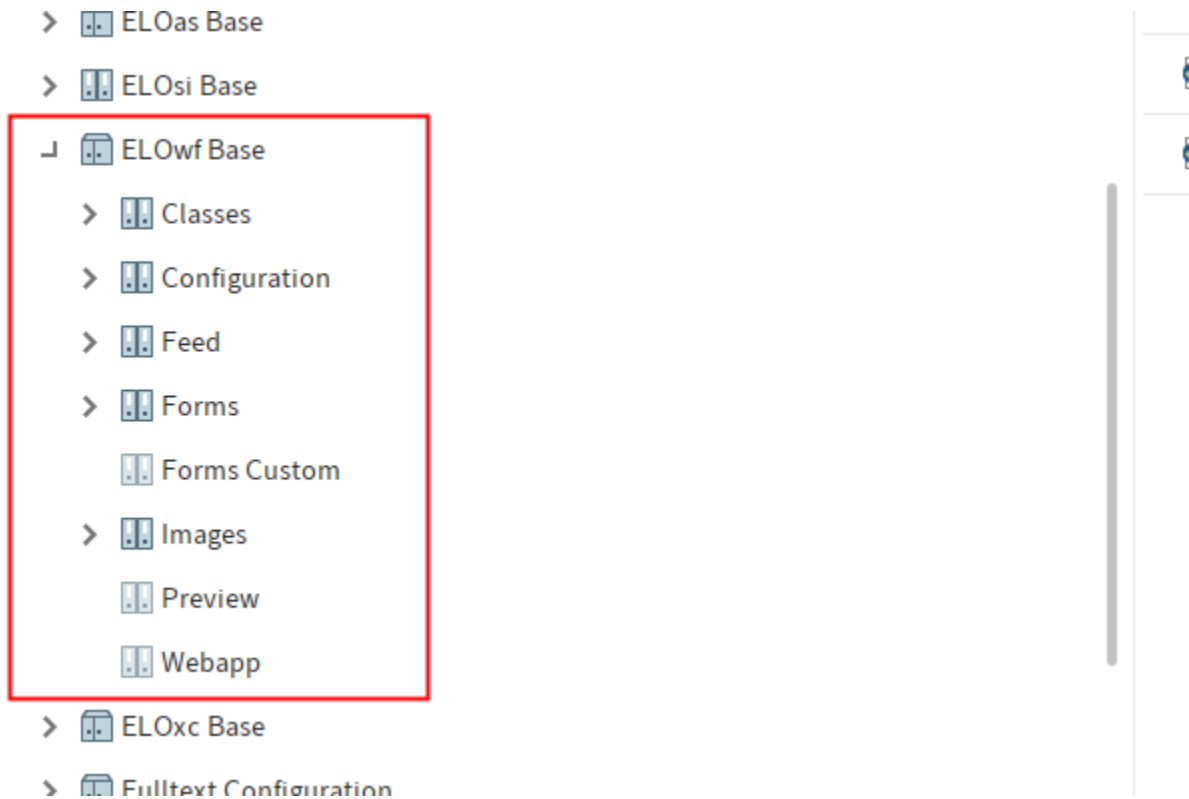
The selected CSS file is embedded in the header part of the form.

8. Select *OK* to save the changes.

Structure in ELO

All form data is stored in ELO under:

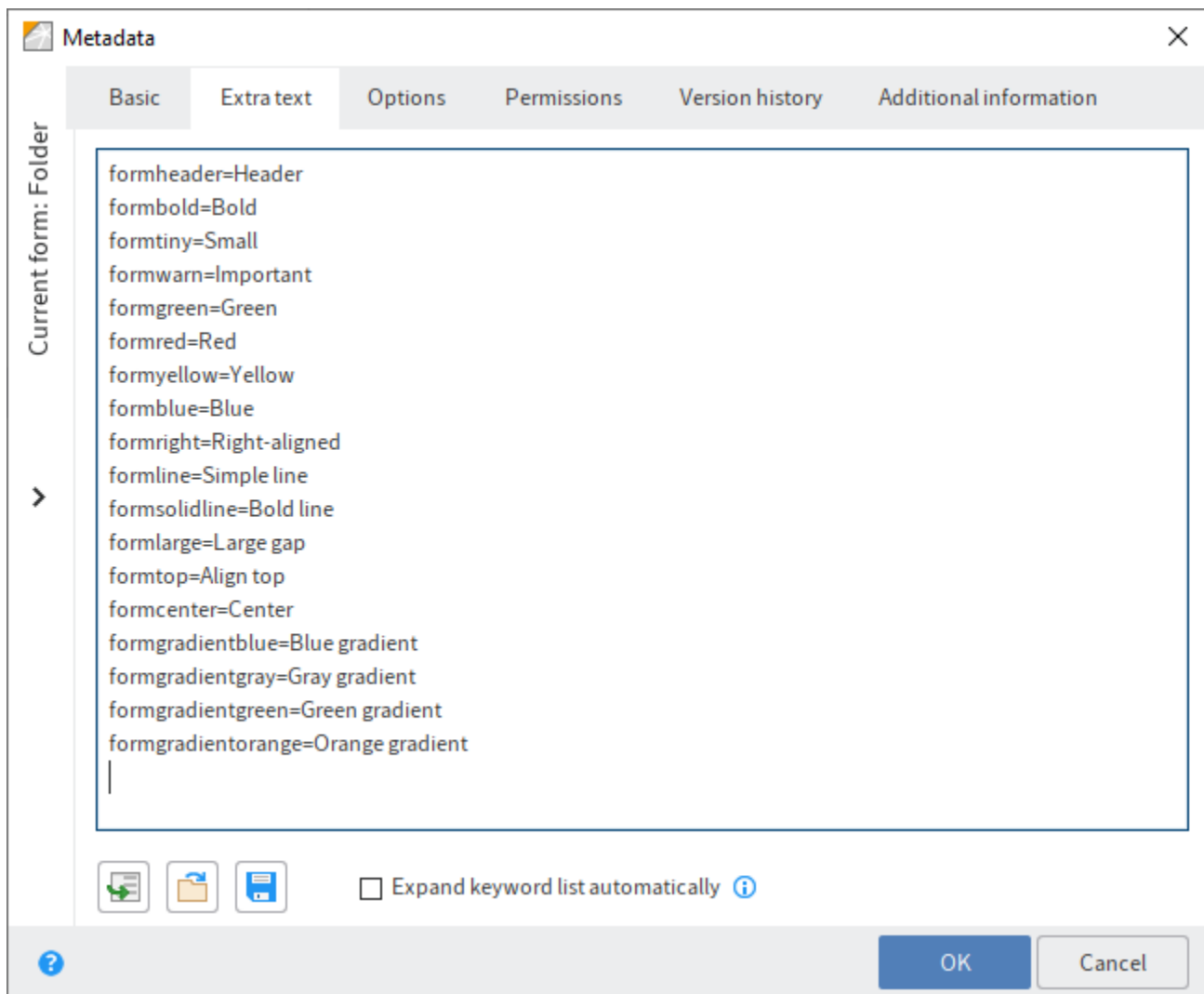
Administration//ELOwf Base



'Classes' folder

The CSS style names for displaying the form are stored in the extra text of the *Classes* folder.

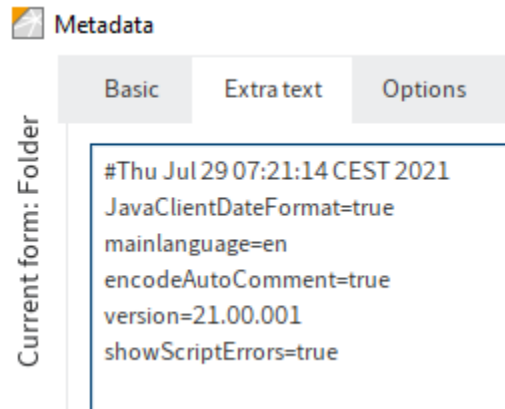
Style names



You can add as many entries as you want to this list. You must define the respective style using custom CSS files or the header part of the form.

Save custom CSS files to the *Classes* folder. You can load them to forms from here. Refer to the section *Custom styles* for more information (see above).

'Configuration' folder



The extra text tab of the *Configuration* folder contains configuration information.

Frame

In addition, the *Configuration* folder contains the default HTML document *Frame* that serves as a framework for all forms. If required, you can create your own frame document for individual forms. However, this document will then be used as the default for new forms.

Import and export

If you are using a customized frame file, make sure that you also transfer it when committing to another repository. You need to replace the frame file in the target repository with the frame file you imported.

'Forms' folder

The *Forms* folder is used by the form designer to create child folders with forms and templates. Under normal circumstances, you will not need to make manual changes here.

Information

After you have made changes, you must select *Refresh* in the form designer to reload the saved forms.

'Forms Custom' folder

The *Forms Custom* folder is intended for the ELO Business Solutions. This is where you store copies of ELO Business Solution (partial) forms that share the same name. You can then edit the copies. This way, your custom settings will be retained in the copies when you update the ELO Business Solution. If there are forms or partial forms in the *Forms Custom* folder, ELO always uses the settings for the copies instead of those for the original forms.

'Images' folder

In the *Images* folder, you can store your own image files, which are then available through the *Image* function in the editor.

'Preview' folder

The *Preview* folder is used for previewing forms in the form designer. As forms always have to be linked to an ELO object for you to be able to view them, the *Preview* folder is used as a wildcard object for the preview.

'Webapp' folder

You can store your own script files or HTML files in the *Webapp* folder. These files are copied to the *Webapp* server directory when you start the program or refresh the repository and can be used when you are working in the form.

Advanced functions

Introduction

This chapter deals with advanced topics on workflows, forms, and the ELO Web Forms Services (ELOwf) module.

This chapter contains the following topics:

- Scripts in the workflow
- Script events and global functions
- End workflows

Using scripts

You can use scripts to integrate additional functions and automated processes into workflows.

Information

This manual does not provide basic information about scripting in general.

Different kinds of scripts can be integrated into a workflow:

- Form header scripts
- Start scripts
- End scripts
- Action scripts

Form header scripts

Form header scripts can be integrated into form-based workflows. You enter form header scripts in the header data of the form. Use the *Form header scripts* component for this.

- Script language: JavaScript



You can save any number of additional script functions here. To ensure that the scripts do not conflict with the default functions, you should add a prefix to the name of your custom functions (e.g. fctReadValue instead of Read Value).

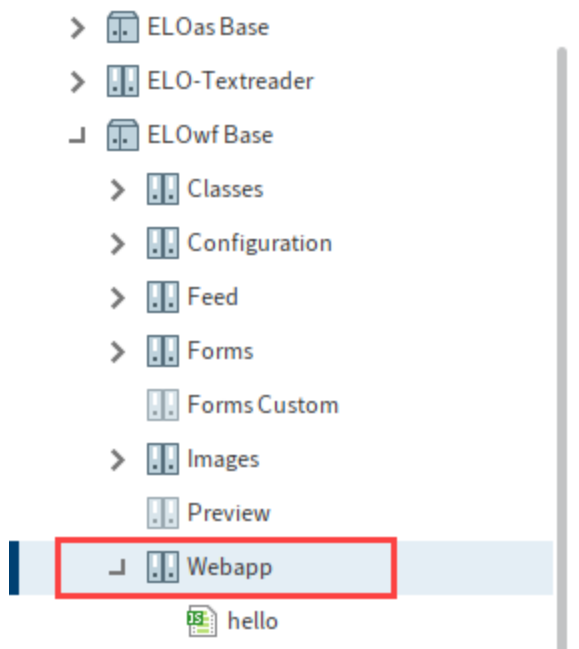
Please note

Functions (buttons and validation) that are to be triggered directly from the form must start with the prefix JS_.

JavaScript files

If you want to use own or third-party JavaScript libraries in the form, you should store these under:

Administration//ELOwf Base//Webapp



Then, restart ELOwf and import the script to the *Form header scripts* component of the form.

```
<script type="text/javascript" src="hello.js"></script>
```

Start/End scripts

Start scripts and end scripts can be integrated into workflow nodes via the corresponding fields. Start scripts and end scripts are run via the ELO Indexserver.

- Script language: JavaScript
- Character encoding: UTF-8

There are two storage locations for the scripts in ELO:

Scripts for all Indexservers used: *Administration//IndexServer Scripting Base//_ALL*

Scripts for only one Indexserver: *Administration//IndexServer Scripting Base//<Indexserver name>*

Information

New scripts are only available after restarting the respective ELO Indexserver.

Start scripts

Start scripts are run as soon as a workflow reaches the respective workflow node.

Start scripts must contain the following functions so that the ELO Java Client can recognize them:

onEnterNode

```
onEnterNode(ci, userId, workflow, nodeId)
```

The function will be run by the Indexserver when a workflow node is entered. The following parameters are transferred:

- ci: Information on the language, country, and ticket (=ClientInfo)
- userId: ID of the current account
- workflow: The current workflow
- nodeId: ID of the respective node

End scripts

Use end scripts to define an action that will run when forwarding the workflow.

For the ELO Java Client to recognize end scripts, they must contain the following functions:

onExitNode

```
onExitNode(ci, userId, workflow, nodeId)
```

The parameters correspond to the parameters for start scripts.

Action scripts

Form ⓘ

Action buttons ⓘ

Fields

Script properties ⓘ

Use the *Action buttons* field (in user nodes) to integrate up to five action scripts into a workflow node. Open the *Action scripts* dialog box with the *Select* button (next to the *Action buttons* field). Select the desired scripts.

Information

If you want to use action buttons, you have to enter at least two action scripts.

Forward workflow

Finish editing the current node

Action buttons: New Excel document, New Word document

Information on the current node

Workflow step: Concept

Comments ⓘ

Forward

Select the next workflow step:

→ Complete concept

Cancel

Action buttons for the respective node appear as additional buttons in the *Forward workflow* dialog box.

The following conditions must be met for the ELO Java Client to recognize and run action scripts:

- Script language: JavaScript
- Character encoding: UTF-8

The action scripts must be saved under *Administration//Java Client Scripting Base* in ELO.

The action script should include the following functions. Replace the wildcards (such as <NAME>):

Action

```
function cfb<NAME>Start(){
}
```

Button label

```
function cfb<NAME>Name(){
    return "<LABEL>";
}
```

Tooltip

```
function cfb<NAME>Tooltip(){
    return "<TOOLTIP>";
}
```

Once the script has been saved at the specified location in ELO, you need to reload the scripts. Use the keyboard shortcut CTRL + ALT + R for this.

Example

The following is an example of an action script for an action button. The action in question here opens a blank Microsoft Excel document. The required Jacob classes (Jacob = JavaCOM Bridge) are imported via the first lines of the script.

```
//Import classes
var importNames = JavaImporter();
importNames.importPackage(Packages.com.ms.com);
importNames.importPackage(Packages.com.ms.activeX);
importClass(Packages.com.jacob.activeX.ActiveXComponent);
importClass(Packages.com.jacob.com.Dispatch);

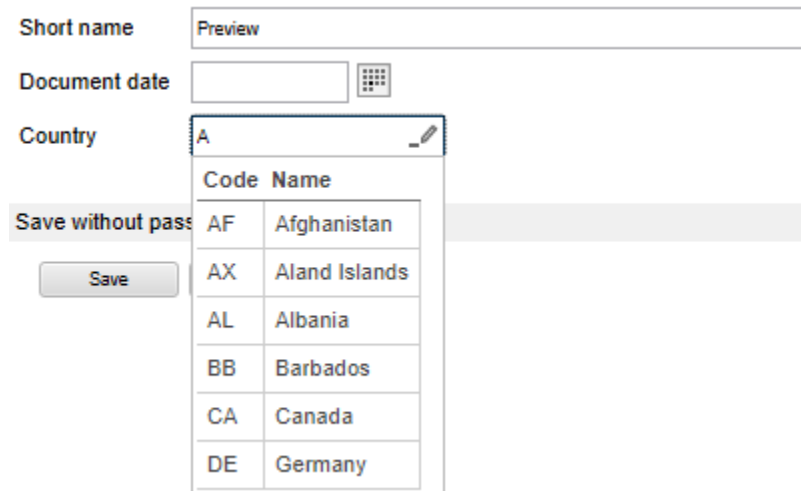
//Open Excel
function cfbOpenExcelStart(){
    var xl = new ActiveXComponent("Excel.Application");
    Dispatch.put(xl, "Visible", 1);
}

//Button label
function cfbOpenExcelName(){
    return "New Excel document";
}

//Button tooltip
function cfbOpenExcelTooltip(){
    return "Open a new document in Microsoft Excel";
}
```

Dynamic keyword list

A dynamic keyword field is a field that can display several columns of content, typically dynamically filtered depending on one or more other field values. It looks like this:



The screenshot shows a form with the following fields and controls:

- Short name:** A text input field containing the word "Preview".
- Document date:** A date selection field with a calendar icon.
- Country:** A dropdown menu currently showing "A". Below it, a dynamic keyword list is displayed as a table with two columns: "Code" and "Name".
- Save without password:** A button.
- Save:** A button.


Code	Name
AF	Afghanistan
AX	Aland Islands
AL	Albania
BB	Barbados
CA	Canada
DE	Germany

Dynamic keyword lists are a relatively advanced ELO feature that requires scripting knowledge. They are available in the ELO Java Client and the ELO Web Client.

They can be used to:

- Display multiple columns of data
- Retrieve content dynamically (e.g. external databases, computed on the fly, ELO scripts...)
- Filter data based not only on the provided field, but also on other arbitrary fields (e.g. what the user has typed and a category that the user previously selected)
- Complete many other fields upon selecting an item (including read-only fields)

The use of a dynamic keywording list is defined directly in the metadata form:

Field group	<input type="text" value="DYN_LIST"/>	
Name	<input type="text" value="Countries"/>	
Translation variable	<input type="text" value="Translation variable"/>	
Display mode	<input checked="" type="radio"/> Normal access <input type="radio"/> Read-only <input type="radio"/> Hidden	

- > Input
- > Properties
- ▼ Keyword list

Edit keyword list

- Only entries from a keyword list allowed
- Translated keyword list

Dynamic keyword list

1. Create a field template.
2. Enter a value in the *Field group* field.

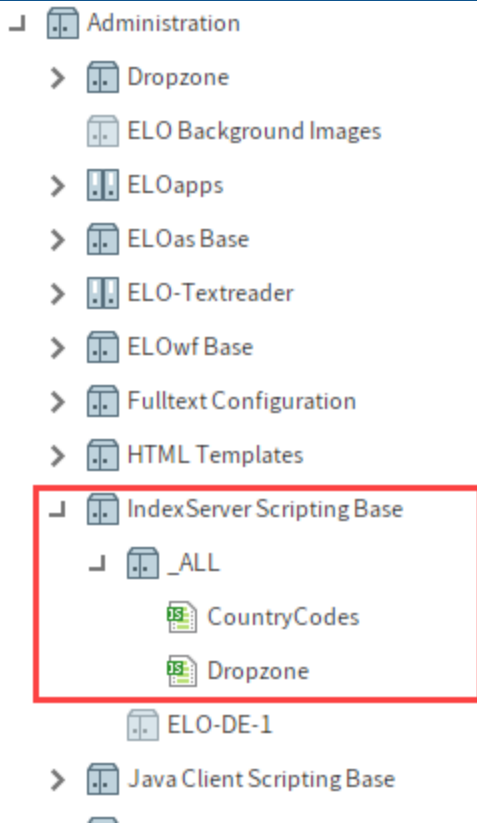
You need this value later on in the form designer.

In this example, the value "DYN_LIST" is used in the *Field group* field of the field template.

3. In the *Keyword list* area, enter the name of a script file in the *Dynamic keyword list* field.

The corresponding script must exist in the following folder:

Administration//IndexServer Scripting Base.



In this example, the script *CountryCodes* is entered. You will find a short version of the script in the following section [Sample script](#).

- To use the list in a form, you have to enter the value from the *Field group* of the field template in the form designer in the *Group name* field.

In this example, that is the value "DYN_LIST".

Properties of the selected cell

Field type	Input
Text	<input type="text"/>
Variable name	IX_GRP_DYN_LIST <input type="button" value="v"/>
Keyword list	Keyword <input type="button" value="v"/>
Group name	DYN_LIST
	<input checked="" type="checkbox"/> Autofill
	<input type="checkbox"/> Only list values allowed

Sample script

The *CountryCodes* script used in the example is a script that provides a static list of country codes and names:

```

importPackage(Packages.de.elo.ix.jscript);
importPackage(Packages.de.elo.ix.scripting);

function getDataIterator() {
    try {
        log.info("getDataIterator(SimpleDatabaseQuery)");
        return new DynamicKeywordDataProvider(new CountryCodes());
    } finally {
        log.info(")getDataIterator");
    }
}

function CountryCodes() {
    var index = 0;
    var results = [];

    /* Utility function to filter a list of countries */
    this.filterCountries = function(filter) {
        log.info("filter: " + filter)
        filter = filter.toLowerCase()
        results = [];
        for (var i=0; i<isoCountries.length; i++){
            if (isoCountries[i].cname.toLowerCase().indexOf(filter) >= 0) {
                results.push([isoCountries[i].ccode, isoCountries[i].cname]);
            }
        }
        log.info("Nach Filter: " + results.length);
    }

    /* Called when initializing a dynamic list (via declaration in the metadata form) */

    this.open = function(ec, sord, focus) {
        log.info("open");

        this.target = focus;

        /*In this case the first field in the form is selected, but ideally a different one should be selected*/

        var filter = sord.objKeys[0].data[0] || "";

        this.filterCountries(filter);
    }

    /* Called when initializing a dynamic list via EL0wf "Dyn. keywording map" field */

```

```
this.openMap = function(ec, map, focus) {
    log.info("openMap");
    log.info(JSON.stringify(map));

    this.target = focus;
    var filter = map[focus] || "";

    this.filterCountries(filter);
}

this.close = function() {
    log.info("close()");
}

this.getNextRow = function() {
    var row = results[index];
    index++;
    log.info("getNextRow(" + row + ")");
    return row;
}

/* The name of the columns */

this.getHeader = function() {
    log.info("getHeader()");
    return ["Code", "Name"];
}

/* The selected item's target */

this.getKeyNames = function() {
    log.info("getKeyNames()");
    return [this.focus, "A_DIFFERENT_FIELD"];
}

this.hasMoreRows = function() {
    log.info("hasMoreRows()");
    return (index < results.length - 1);
}

/* Either return an error message or leave it empty on success */

this.getMessage = function() {
    log.info("getMessage()");
    return "";
}
```



```

    this.getTitle = function() {
        log.info("getTitle()");
        return "Country Codes";
    }
}

var isoCountries = [

    {'ccode' : 'AF', 'cname' : 'Afghanistan'},
    {'ccode' : 'AX', 'cname' : 'Aland Islands'},
    {'ccode' : 'AL', 'cname' : 'Albania'},
    /* ... */
    {'ccode' : 'ZW', 'cname' : 'Zimbabwe'}
];

```

Read more:

- [ELO Indexserver programming guide > Dynamic keyword lists](#)

Dynamic keyword map

A dynamic keyword map is similar but is based on MAP fields.

Properties of the selected cell

Field type	Input
Text	<input type="text"/>
Variable name	IX_MAP_COUNTRY <input type="button" value="v"/>
Keyword list	Dynamic Keyword Map <input type="button" value="v"/>
Script name	CountryCodes
Filters	<input type="text"/>
	<input checked="" type="checkbox"/> Autofill
	<input type="checkbox"/> Only list values allowed

1. In the form designer, select an input field.
2. From the *Keyword list* drop-down menu, select the entry *Dynamic Keyword Map*.
3. Enter the target script in the *Script name* field.

Notes

When an entry is selected in a dynamic keyword list, the following event function is called in the header script:

```
onDynListItemSelected(item)
```

Please note

It is not possible to combine the *Autofill* option with the *Only list values allowed* option for dynamic keyword lists. The main reason for this is that dynamic keyword lists may depend on several input fields and may also modify multiple input fields. It is not currently possible to support multi-field validation.

All lists can also be triggered within the scripts:

```
/**
 * Calls a specific rule in ELOas.
 */
function $listAs(scriptName, param2, param3, onSuccess, onFailure) {
/**
 * $listKw("IX_GRP_DYN_FIELD", ...) will retrieve the data from the keyword
 * list defined in the specified field
 */

function $listKw(swlid, onSuccess, onFailure) {

/**
 * $listDyn("MyScript", "foo", ["bar"], ...) will retrieve the data from the
 * corresponding 'IndexServer Scripting Base' script.
 * The script will be invoked with "foo" as focus name and {"foo": ...,
 * "bar": ...} as map data, also replacing the wildcards "{i}" and "{*}"
 */
function $listDyn(scriptName, focusfield, filterfields, onSuccess, onFailure) {
```

Calculation with scripts

If you get values from fields using scripts, always use the function `$num(...)` and not `$val(...)`.

The first function returns a float, while the second function returns a string in "ELO format", i.e. no thousands separator and a comma as the decimal separator, regardless of the requested language.

If the input contains the value "12,345.67", `$val("IX\...")` would return the String "12345,67".

In `ELO_PARAMS`, number and amount values are always stored as strings in ELO format as well.

Rounding numbers

The form is executed within JavaScript without context knowledge, so it has no way of knowing what the number field is for. If a number field has more decimal places than are specified, we cannot expect the correct behavior in all cases.

A user mapping critical processes should refrain from designating a formula and instead come up with a method for rounding in the *inputChanged* event.

Save and forward validation

Since version 10.01, a check is performed when saving data that determines whether changes were made on the server since the form was loaded (and the data read out).

This could occur, for example, if user A changes the metadata while user B is working on the form. If user B has loaded the form before the changes are made, that user will not see these changes, which up to now were simply overwritten.

If the system detects that data has changed, a dialog box notifies the user. The responsible user can then confirm whether to overwrite the data or to cancel saving and forwarding.

In the script, entering

```
ELO.Configuration.ForceSave = true;
```

sets the previous behavior and all changes are overwritten directly without a prompt.

Print

Please note

The following features are only applied if the user prints a document using the *Print* button. They do not apply if the user presses CTRL+P or prints via a browser menu.

Auto-expand text areas

When printing, you can set the following flag to see the full content of text areas:

```
ELO.Configuration.PrintExpandTextarea = true;
```

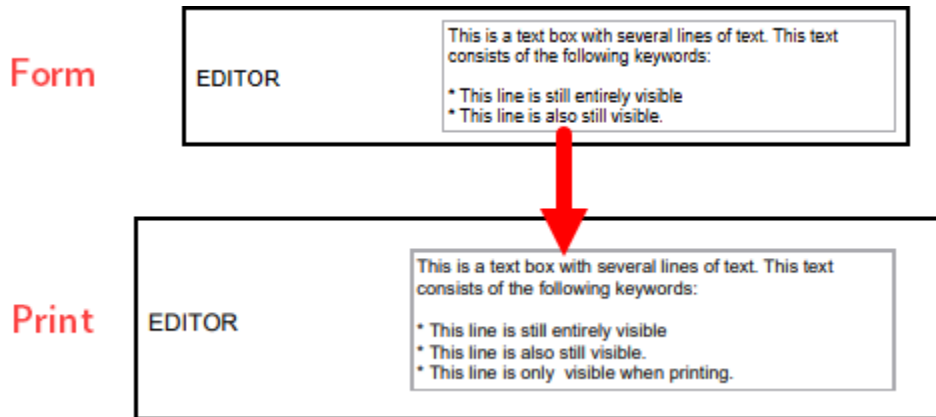
Edit form header scripts

```

1 <script type='text/javascript'>
2
3   function onInit() {
4   }
5
6   function inputChanged(source) {
7     if (!source) {
8       ELO.Configuration.PrintExpandTextarea = true;
9     }
10  }
11

```

Place the flag in the *inputChanged()* event function.



This makes all text areas sufficiently larger.

Print all tabs

If you want to print all tabs by default, you can set following flag:

```
ELO.Configuration.PrintAllTabs = true;
```

Edit form header scripts

```

1 <script type='text/javascript'>
2
3   function onInit() {
4   }
5
6   function inputChanged(source) {
7     if (!source) {
8       ELO.Configuration.PrintAllTabs = true;
9     }
10  }
11

```

Place the flag in the *inputChanged()* event function.

Form

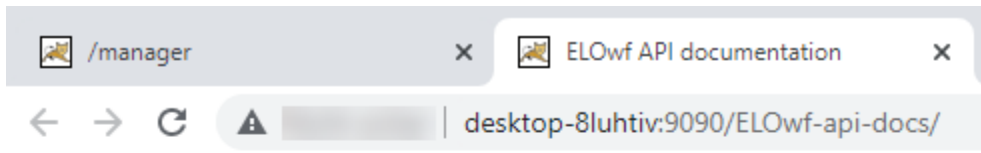
A small thumbnail of a 'Material order form'. At the top, there is a header bar with three tabs: 'Order', 'Approval', and 'Completion'. The 'Order' tab is currently selected. Below the header, the text 'Material order form' is displayed in a large, bold font.

A large, detailed view of the 'Material order form' showing three distinct stages, each separated by a horizontal line with a decorative zigzag pattern. Each stage has a tab at the top and the text 'Material order form' below it. The stages are: 1. 'Order' stage with a tab labeled 'Order'. 2. 'Approval' stage with a tab labeled 'Approval' and a 'Print' button located just below the zigzag line. 3. 'Completion' stage with a tab labeled 'Completion'.

Events and global functions

You can use predefined events and global functions when creating scripts. For a detailed list of predefined events and global functions, see the *ELO Web Forms API Documentation*.

The *ELO Web Forms API Documentation* is available as an additional package and can be installed on the ELO server at a later point in time. Extract the ZIP file matching the installed version of ELOwf and copy it to one of the ELO servers in the *webapps* folder.



ELO Forms API documentation

[ELO Forms API documentation](#)

ELO Apps API documentation

[ELO Apps API documentation](#)

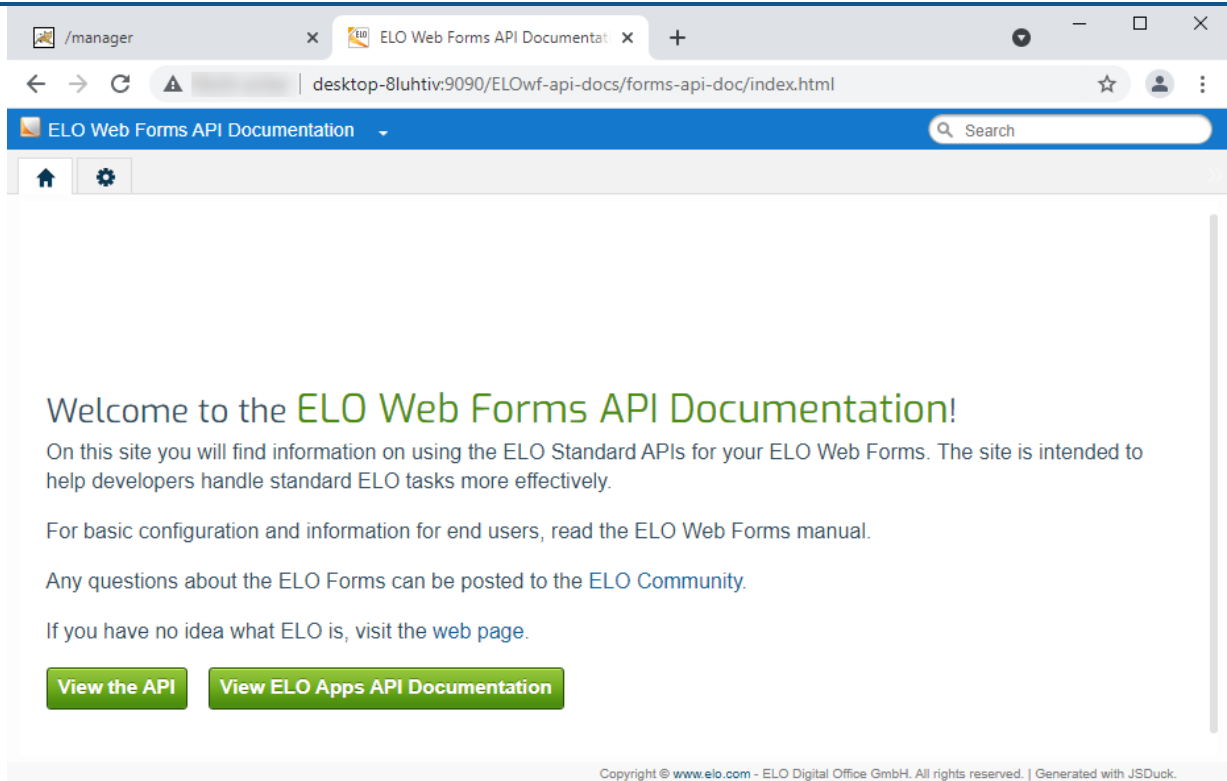
1. Open the ELOwf API documentation start screen via a URL with the following structure:

```
http(s)://<server>:<port>/ELOwf-api-docs/index.html
```

2. Open the *ELO Forms API documentation* link.

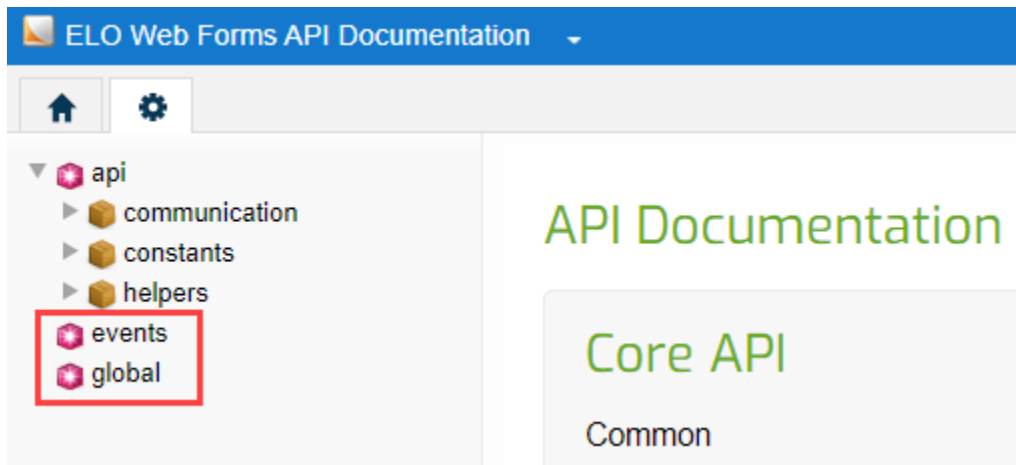
Alternative: To go straight to the *ELO Web Forms API documentation* page, enter a URL in the browser using the following syntax:

```
http(s)://<server>:<port>/ELOwf-api-docs/forms-api-doc/index.html
```



The *ELO Web Forms API Documentation* page opens.

3. Open *View the API*.



The *API Documentation* tab (gearwheel icon) appears.

Explanations of the events can be found under the menu item *events*.

Explanations of the global functions can be found under the menu item *global*.

Events

This chapter describes some events fired in ELOwf forms.

onInit()

This event is invoked once when the form is loaded.

inputChanged(source)

This event is invoked each time the user changes the content of an input, for each key stroke. The input field is provided as argument. The event is not triggered when a script changes a value.

saveClicked()

The *saveClicked* event is invoked when a form is saved. More precisely, it is invoked before the form is validated and saved. This can be used to either perform additional validation or to set other form variables before saving.

Example:

```
function saveClicked() {
    if( $num("IX_GRP_PRICE") > 10000 && $val("IX_GRP_TYPE") == "CHEAP"){
        eloAlert("Sorry, that's too expensive!");
        return false; // Do not save and cancel
    }
    else {
        return true; // ok, proceed with validation and saving
    }
}
```

Since ELOwf 10.1, asynchronous processes can also be performed by returning promises:

```
function doSomething(resolve, reject) {

    // retrieve or verify some data asynchronously
    // call 'resolve()' when successfully finished
    // or 'reject()' if the process should be aborted
}

function saveClicked() {
    return new Promise(doSomething);
}
```

The promise is a delayed result that is used to continue the process when it is resolved or rejected.

nextClicked(id)

This event works exactly the same way as *saveClicked* but is invoked when the form is forwarded to the next workflow node.

The following values/results can occur:

- true: Workflow is forwarded
- false: Workflow is not forwarded
- Promise: Trigger is delayed

The only difference between the way this event and the previous event functions is its additional parameter `id`, which references the ID of the workflow node to which the workflow is being forwarded.

Instead of the ID, it is also possible to recover the internal node name:

```
var nextNodeName;
for (var i=1; i<20; i+=1) {
  if (ELO_PARAMS["NEXT_" + i] &&
      ELO_PARAMS["NEXT_" + i].indexOf(String(id) + "\t") === 0){
    nextNodeName = ELO_PARAMS["NEXT_" + i].split("\t")[1];
  }
}
```

Iteration is performed on all potential successor nodes (max. 20), which are then compared with the `id`.

The extracted node name is the technical or translated name of the node. If using a localization key here, the localization key is specified in `ELO_PARAMS["KEY_NEXT_" + i]`, which is more suitable for comparison as it is identical in all languages.

A differentiation is also made between the name of the successor node and the label where the current node is shown. Script nodes can thus be bypassed for the user. If these fields are used in the workflow, the translated display name for the node is shown in `ELO_PARAMS["LABEL_NEXT_" + i]` along with the localization key under `ELO_PARAMS["LABEL_KEY_NEXT_" + i]`. You will have to check that these additional entries exist, as they are optional.

Global functions

Examples of global functions:

\$val(name)

Use this function to query the content of an input field with the name `name`. Example: `var name = $val("IX_GRP_NAME");`

\$num(name)

As with the function `$val`, this function returns the content of the field with the name `name`. In this case, however, it is returned as a numeric value and not as text. Example: `var vat = $num("IX_GRP_VAT");`

\$update(name, value, force)

This function enters the specified value into the *name* input field and then invokes the validation. The validation is needed to check if all inputs are permissible. If necessary, an error message will be displayed and the window refreshed. You can use the parameter *force* to force the value to be saved, even if no corresponding field exists in the metadata. To do this, you must transfer the value `true` to the parameter *force*.

End workflows

Workflows are usually completed when the last node is finished. End nodes can also be used for terminating a workflow.

In some cases, however, you may still need to terminate a workflow manually.

Please note

Once you have terminated a workflow, it cannot be reversed.

1. Select *Workflow overview* (*Ribbon > Organize > Overviews*).

The *Workflow overview* dialog box opens.

Optional: You can set additional filters if required.

1. Select the workflow you want to terminate.

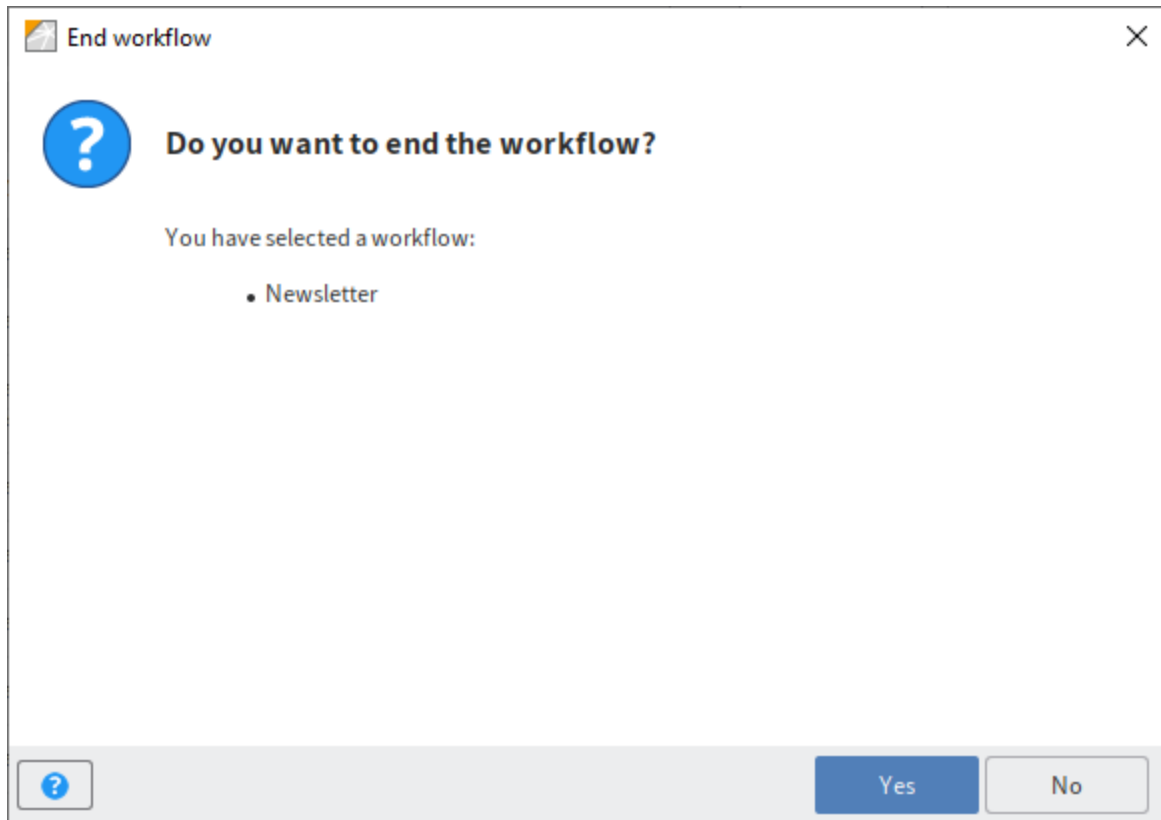
The screenshot shows the 'Workflow overview' dialog box. It has a 'Status' section with radio buttons for 'Active' (selected), 'Completed', and 'All workflows', and checkboxes for 'Passed deadlines only' and 'Load fields'. There is also a 'User' dropdown menu set to 'All users and groups'. Below this is a table of workflows with columns for Name, Template, Type, and Start date. A context menu is open over the 'End workflow' option, which is highlighted with a red box. The context menu options are: Go to, Metadata, Extend deadline, PDF output, Save table to clipboard, Show subworkflows, End workflow, and Delete workflow permanently.

Name	Template	Type	Start date
Newsletter			Today 11:34 AM
AddNoteConfirm			Sep 14, 2021, 10:..
AddNoteConfirm			Sep 14, 2021, 9:3.
Approval workflo			Sep 14, 2021, 9:3.
Material order for			Sep 14, 2021, 9:3.
Incoming Invoice			Sep 14, 2021, 9:3.
AddNoteConfirm			Sep 14, 2021, 9:3.
Newsletter			Sep 14, 2021, 9:2.
Order			Sep 14, 2021, 9:2.

- 2.

Open the context menu.

3. Select *End workflow*.



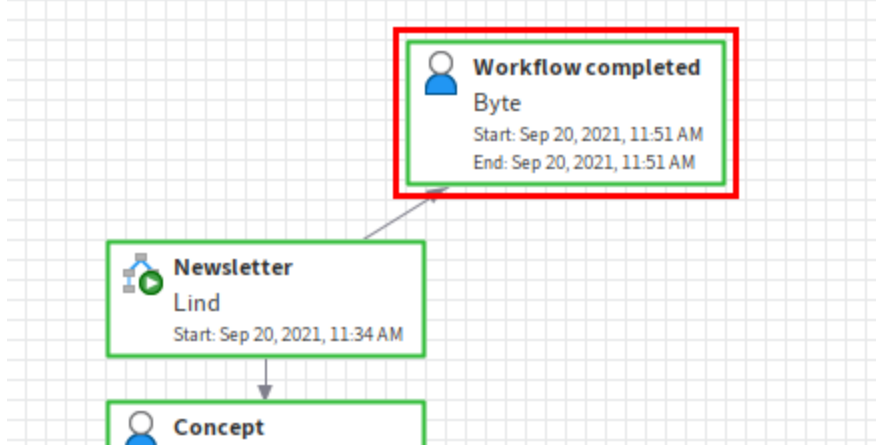
The *End workflow* dialog box opens.

4. Click *Yes* to confirm the dialog box.

The selected workflow will be terminated.

In the *Workflow overview* dialog box, you can set the *Completed* filter so that you only see workflows that have been completed.

The graphical view now contains an additional user node next to the start node, with the name of the person who terminated the workflow.



In addition, the date of termination is recorded in the user node.