ELO

Document Management and Archiving Software

ELO Java Client Workflow

April 2017

ELO Digital Office GmbH

http://www.elo.com

Copyright © 2017 ELO Digital Office GmbH

All rights reserved
# Table of Contents

**Introduction** .......................................................................................................................... 5

Organization ................................................................................................................................. 6
Conventions ................................................................................................................................. 7

**Basics** .................................................................................................................................. 9

Types of workflows .................................................................................................................... 10

**Ad hoc workflow** ................................................................................................................. 12

Creating ad hoc workflows ......................................................................................................... 13

**Default workflow** ................................................................................................................ 20

The workflow designer ................................................................................................................ 21

Toolbar ....................................................................................................................................... 23
Creating workflow templates ...................................................................................................... 27
Editing nodes ............................................................................................................................... 37
Editing and managing templates .................................................................................................. 54

**Form-based workflow** .......................................................................................................... 66

Basics .......................................................................................................................................... 67
The form designer ......................................................................................................................... 69
Creating new forms ...................................................................................................................... 71

Toolbar ....................................................................................................................................... 73
Creating templates ....................................................................................................................... 89
Attributes of tables ...................................................................................................................... 94
Creating tab groups ..................................................................................................................... 97

Cell properties ............................................................................................................................ 103

Global form settings ................................................................................................................... 117
Integrating a form into a workflow ............................................................................................ 122
Save form data ............................................................................................................................ 127

Validation .................................................................................................................................... 131

Custom styles .............................................................................................................................. 136
Structure in the repository .......................................................................................................... 142

**Advanced functions** ............................................................................................................. 146

Using scripts ............................................................................................................................... 147
End workflows ............................................................................................................................. 157

**Appendix** ............................................................................................................................. 160
Disclaimer of Liability............................................................................. 160

Index........................................................................................................ 161
Welcome to the workflow manual for the ELO Java Client.
This chapter contains the following topics:

- Contents of the manual
- Conventions that apply for the user manual
## Organization

The manual has been divided into the following chapters:

### Introduction
The first chapter contains a general introduction to workflows, describes the objectives of the user manual and provides an overview of the contents.

### Basics
The second chapter deals with the basics of creating workflows. You will learn what types of workflows exist and what functions these workflows have.

### Ad hoc workflow
The third chapter deals with ad hoc workflows.

### Default workflow
In the fourth chapter you will learn what a default workflow is and how you can make optimal use of it.

### Form-based workflow
The fifth chapter deals with the topic of form-based workflows and the use of the ELO form designer. The form-based workflow editor is used for creating forms, which can be embedded in workflows.

### Advanced functions
The sixth chapter describes the advanced workflow functions, such as the use of scripts.

### Appendix
The appendix includes the Disclaimer of Liability and the index.
Conventions

This manual describes a large number of interactions, program dialog boxes, menus and interfaces. The following conventions apply for this:

Notes

Important information is displayed in a gray box with an information icon. The following types of notes exist:

- **Information**: This is additional information that makes working with ELO easier.

- **Please note**: Take this note into account to ensure the smooth operation of your program.

- **Warning**: It is essential that you adhere to this note to avoid significant restrictions in the operation of your program.

Italics

The names of menus, options, dialog boxes, folders, references to chapters, paths, and file extensions are written in italics.

Example: Click *ELO > Configuration* and choose the *Display* option.

Keyboard shortcuts

Shortcuts are written in UPPER CASE. Keys pressed at the same time are indicated with a plus icon (+).

Example: CTRL + C

Bold

*Notes* and **highlighted information** are marked in **bold** in the user manual. This also applies for the words *alternatively, optional*, and the names of examples.

Courier

Program code, program outputs, inputs and scripts are written in Courier font.

Example: *MsgBox "Hello world!"

Paths

This manual differentiates between three types of paths:
Navigation paths: These paths describe the order of buttons and menu items you click through to use certain functions. You can recognize navigation paths by the pointy brackets (>) and the italic font.

Example: Click the *ELO > Configuration > Display > Repository layout*.

Filing paths: These paths describe filing paths within the ELO repository. You can recognize filing paths by the pilcrow sign (¶).

Example: Save the document under ¶ Filing ¶ Year ¶ Month.

Document and filing paths: These paths describe storage locations for files within the operating system. We indicate this by using backslashes (\).

For example: Save the document under C:\Documents\Vacation.

Target audience

The manual is intended for *advanced users* who are already familiar with the basic functions of the ELO Java Client and who want to learn more about workflows.
Basics

This chapter contains the following topics:

• Types of workflows
Types of workflows

ELO offers the following types of workflows:

- Ad hoc workflows
- Default workflows
- Subworkflows

**Ad hoc workflows**

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

- **Approval** means that a workflow can only be ended if all included users have marked the assigned task as completed.

- **Inform/Distribute** means that the participants of the workflow must confirm the receipt of the workflow. The information associated with the workflow (such as in form of a document) is considered to have been received.

You can find further information regarding the different types of ad hoc workflows in the *Ad hoc workflow* chapter.

**Default workflows**

Default workflows cover a large range of company processes. In ELO, you use workflow templates to do so. Configure these templates according to the requirements of the respective process. The workflow templates help to provide standardized templates for all participants.
Sample invoice approval

Fig. 1: 'Incoming invoice' workflow

If a company receives an invoice, for example, certain processes must be adhered to. In ELO, the responsible employee selects the corresponding template and starts the matching workflow in so doing. The invoice goes through the necessary stations (nodes) of the workflow and is approved and paid as soon as the check returns a positive result.

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.
Ad hoc workflow

This chapter contains the following topics:

- Creating ad hoc workflows
Creating ad hoc workflows

Ad hoc workflows are suitable for quickly displaying simple processes. Ad hoc workflows are predefined by ELO and can be set up with relatively little effort. However, this restricts the design options as compared to default workflows.

Proceed as follows to start an ad hoc workflow:

**Step by step**

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

![Fig. 2: 'Ad hoc workflow' button](image)

2. Click the Ad hoc workflow button (ribbon > Tasks).
The **Start ad hoc workflow** dialog box appears.

**Option 1**: Change the name of the ad hoc workflow in the *Name* field as well if you wish. The workflow is displayed under this name.

3. Determine the type of ad hoc workflow. Choose from the following options:
• **Serial approval:** The workflow is passed forward sequentially to the participants. The participants must decide whether or not they approve the workflow. 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. Each participant must approve the workflow.

• **Serial inform/distribute:** 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 user does not acknowledge the workflow, the workflow is not passed forward to the following participants.

• **Parallel inform/distribute:** The workflow will be sent to all participants at once. The participants should confirm receipt of the workflow. After completion, the corresponding notification is sent to the selected recipient.

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

   Click the corresponding suggestion to select a user or a group.
Alternatively: 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 click the Expand button, the group will not be displayed as a group. Instead, the individual members of the group are listed.

Information: If the group is selected, one member of the group can accept and edit the workflow. If you expand the group, every member of the group can accept and edit the workflow.

Option 2: To remove a user or group from the list, click the X icon next to the corresponding entry.
**Option 3**: The order of the participants in the distribution list can be changed. To do so, drag the individual users to where you would like to have them or use the *Move up* (small arrow icon pointing up) or *Move down* (small arrow icon pointing down) buttons.

**Option 4**: Change the text in the *Job instructions* field as well if you wish. The respective job instructions apply for the user/group currently selected. Multiple selection is possible. The job instructions can have a maximum of 128 characters.
Option 5: You can also set the maximum amount of time the ad hoc workflow may remain in the possession of the selected user/group. Click the Escalation management button.

Option 6: Select a recipient for the success notification when completing the workflow. To do so, expand the Closing options area and click the Select user button.

Option 7: Expand the Closing options area and change the message that is shown to the recipient when the workflow is successfully completed in the Success notification field. The text also appears as a button when passing forward the workflow.

Option 8: Expand the Closing 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 passing forward the workflow.
Information: For *Inform/Distribute* workflows, the field *Not approved notification* is disabled. No message is sent as long as at least one participant still has not taken note of the workflow.

**Option 9** (only applies for *Parallel approval* workflows): Expand the _Closing 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 _Closing options_ area and select a script to be executed when the workflow is completed from the drop-down menu in the _End script row_.

Information: You can find additional information on the topic of _scripts_ in the chapter on *Advanced functions*.

6. Click OK.

**Result**  The *ad hoc workflow* is started.
Default workflow

This chapter contains the following topics:

• The workflow designer
• Creating workflow templates
• Editing nodes
• Editing and managing templates
The workflow designer

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

You can create workflow templates via the Workflow designer dialog box. You can open the workflow designer via ribbon > ELO > 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 view. The workflow is displayed as a diagram in the middle of the dialog. In addition, you will find a button for managing workflow templates in the column.
(2) **Drawing area:** In the drawing area, the workflow templates are displayed in the graphic view. You can see the stations (nodes) and elements of which a workflow consists and you can see how the stations are connected.

(3) **Toolbar:** Here you will find the tools for creating and editing a workflow template.

**Information:** When calling up a workflow template, the *Edit workflow templates* button first appears. Click the *Edit workflow template* button to switch to edit mode.

(4) **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 elected, the *Workflow settings* area is displayed, where you can adjust the general settings.

(5) **Status bar:** Here, you will find the ID of the workflow template, the current version, and the node ID of the currently selected workflow node.
Toolbar

Fig. 11: Workflow designer 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

When the Select option is active, you can drag the nodes of your workflow template and reposition them.

When multiple nodes are selected, the selected nodes can be moved together. Select several nodes by clicking one after another while pressing CTRL or by dragging a frame around the nodes. It is also possible to drag a frame around the desired node by keeping the left mouse button pressed.

Information: The Select function (arrow icon) must be enabled to drag 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 click the nodes. Click the start node first and then click 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 *Determine successor node on timeout* function allows you to connect one node to another, which is automatically called when the set deadline is exceeded.

You can set the deadline in the *Node settings* under *Escalation management > General escalation*. If the deadline is exceeded, the workflow is passed directly on to the respective successor node. Unlike with escalations, the successor node participant does not have to accept the workflow.

**Information:** If you use the *Determine 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 the workflow templates.

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

(5) User node

With user nodes, you determine the stations 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 keywording form with specific values, for example, to check the amount of an invoice.

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

(7) Distribution node

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

(8) Collection node

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

If an invoice needs to be approved by two employees, for example, the invoice is not passed forward until both employees have approved the invoice.
(9) Cycle node

In workflows, there are processes 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 and an end node. 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] ]. These five characters must be used for 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 decision as to whether a cycle is passed a second time is made in the end node of the cycle. 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 identifier of the second server must be entered in the server transfer node.

After synchronizing the data to the second server, the workflow can be processed further. When the workflow is complete, the particular data will be transferred back to the first repository.

(11) Subworkflow

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

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

(12) End node

The end node is used to define a unique end point of a workflow.
It is not mandatory to use an end node. Without an end node, a workflow ends when there are no more open 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, but the workflow remains active since there are still open nodes. An end node is required to fully complete the workflow.

Additional buttons in the toolbar

![Additional buttons in the toolbar](image)

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

13) Minimized view

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

14) Use grid

This function serves to activate or deactivate a grid in the background of the drawing area.

15) Open form designer

This function opens the form designer to create forms for integration into a form-based workflow.

**Information:** You can find additional information on this topic in the chapter on *Form-based workflows.*
Creating workflow templates

Use the workflow designer to create a new workflow template. Processes are displayed via workflow templates. A workflow template must exist when starting a default workflow.

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

Creating a template

1. Click the ribbon > ELO > Workflow designer.

2. The Workflow designer dialog box opens.

3. Click New.
The Create new template dialog box appears.

4. Enter a name for the new workflow template. In our example Newsletter. Confirm by clicking OK.

**Start node set**

The new template is now available in the Templates column and selected. The start node appears in the drawing area.

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. Referring back to the start node is not possible.
You have now created the basic structure of a workflow template. Edit the template according to the requirements of the workflow.

**Starting edit mode**

By default, the workflow designer is in read mode. You can only edit workflow templates in edit mode.

1. Click the *Edit workflow template* button to switch to edit mode.

The buttons for editing workflow templates appear in the toolbar of the workflow designer.

**Editing a 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. Click the start node.

The node is selected.

2. Make the desired settings for the following fields under *Workflow settings*.

**Icon**

Choose a new icon via the *Icon* drop-down menu.

**Transfer to server**

The *Transfer to server* field allows you to enter the repository identifier of a server that the workflow is to be transferred to.
Please note: This option is only necessary when workflows are replicated, i.e. if you have installed the Replication module and are working with more than one broker.

User

![User field]

The default setting for the User field is Owner. Owner means: The person who starts the node will edit all nodes with this setting. It is not possible to make any changes to the start node here.

Priority

![Priority drop-down menu]

The Priority drop-down menu allows you to define the priority of the workflow. You can choose between the following priorities: A (=High), B (=Medium) and C (=Low). This function is useful if a large number of workflows exists and you want to use different weightings in regard to their importance.
Node name

![Node name field](image)

**Internal name**

The name of the workflow template is automatically used for the start node. If needed, change the name in the *Internal name* field.

**Translation variable**

The *Translation variable* field is required if you need the contents of the *Internal name* field in multiple languages. Enter the corresponding key in the respective properties file.

*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**

![Exclude weekends option](image)

When 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 to calculate the workflow duration.

When this option is cleared, Saturdays and Sundays are included in the calculation. All calendar days count for the calculation of 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.

![Fig. 22: Fields for 'General escalation']()

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

**Information:** For *General escalation*, the user is selected via the start node and then applies to all nodes.

If you do not enter a maximum duration, the workflow will not be automatically checked during processing to see if it complies with deadlines.

Escalation B

![Fig. 23: Escalation levels 'B' and 'C']()
The fields under *Escalation B* allow you to define a second escalation level. If this deadline is passed, the corresponding user is informed.

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

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

### Manually start workflow

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

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 Indexserver. Further information on scripts can be found in the *Advanced functions* chapter under *Using scripts.*
Form

In the Form field, you can link part of form (template) to the node. The selected templates are displayed when calling up the node. To open the Workflow form selection dialog box, click Select.

Script properties

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

3. Once you have entered all information, click Apply.

The settings for the start node are saved.

Additional nodes

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 will explain this principle based on the user node. You can find more information on the various node properties in the section on Editing nodes.
Creating user nodes

1. Click the *User node* button.
The cursor then turns into the icon for the user node.
2. Place the cursor where you want to insert a user node.
3. Click the left mouse button.

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

**Linking 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. You cannot link back to the start node.

1. Click the *Establish connection between two nodes* button.
2. Click the start node, i.e. the node where the connection should start.

The node is selected.

3. Click the target node.

A connection is created between the workflow nodes. The arrow icon indicates the direction of the connection.

**Information:** It is possible to connect nodes in both directions. In this case, the connecting line has two arrows.

**Completing the template**

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

1. Click **OK** in the workflow designer.

**Alternative:** To save the template without ending the workflow designer, click **Apply**.

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

**Information:** The *Save* and *Version* functions offer options for saving the template. Refer to the *Editing and managing templates* section for more information on the two functions.
Editing nodes

Depending on the node type, you can determine different account settings.

**Open node**

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

1. Click 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 will describe the most important settings for each node type:

**User node**

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

**General settings**

![User field](Fig. 30: 'User' field)

**User**

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

To choose another user or a group, click *Select*. 
Select second group

If you have selected a group, add another group by pressing Select next to the Add second group button. The two groups will be connected as AND groups. Only users who are members of both groups will receive the workflow.

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 if the member 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, click Link group nodes.
The *Link group nodes* dialog box appears.

2. In the workspace of the workflow designer, click 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, click **Link selected group nodes** (arrow pointing right).

4. Close the dialog box with **OK**.

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

### Comments

You can enter a message for the editor of the workflow node, such as a job instruction or a note on processing to the **Comments** field.
Node name

In the Internal name field, you can enter a name differing from the Internal name field, if necessary. The contents of the Name for forwarding field serve to label the button in the Pass forward workflow dialog box.

If you leave the Name for forwarding field empty, ELO automatically applies the contents of the Internal name field as the label for the forwarding button.

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

Name for forwarding

Translation variable

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

Search options

In the End script field, you can define an action which is carried out when the workflow is passed forward. For example, a status message can be sent to the workflow owner.

End script
**Information**: End scripts are run via the Indexserver. Further information on scripts can be found in the Advanced functions chapter under Using scripts.

**Show after (number of days)**

![Show after field](image)

Use the **Show after (number of days)** field to set up delays. When a delay has been set up, a workflow only appears once the entered days have been passed in the editor's tasks list. The delay becomes effective as soon as the workflow has been forwarded to the node.

**Please note**: The number of days for the delay should not exceed the maximum retention period. Otherwise, the workflow will pass the deadline before it can be edited.

**Only one successor node possible**

![Only one successor node possible](image)

If the option **Only one successor node possible** is enabled, only one node can be selected for forwarding.
Reset successor nodes

If the **Reset successor nodes** option is set, the "completed" status of all the successor nodes of a workflow node is reset if the branch of a workflow repeats as part of a loop. The successor nodes are reset to their original status from the first run-through.

**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.
Press the *Select* button to open the *Workflow form selection* dialog box. There, you can select the desired templates.

In the *Pass workflow forward* 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 calling a document.

To open the *Action scripts* dialog box, click the *Select* button next to the *Action buttons* field. Scripts can be selected in the *Action scripts* dialog box.
The integrated action buttons are displayed in the *Pass workflow forward* dialog box.

**Information:** Certain rules apply for action scripts. Further information on this topic can be found in the *Advanced functions > Using scripts* chapter.

**Index field**

Use the *Index field* field to select which index fields of a keywording form will be used for this node. The user who edits the respective node sees the selected index fields when passing the workflow forward.
Click *Select* next to the *Index field* field to open the *Index fields* dialog box. There you can use a drop-down menu to select the desired keywording form.

The *Groups* field shows you the index field groups used on the keywording form. Use these groups to create connections to the respective index field.

**Information:** You can see which index field is connected with which group in the ELO Administration Console.

**Decision node**

A decision node is used to check the data entered into a keywording form. The data determines the further progress of 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 (>=)
• less than or equal (<=)

**Information:** The node must be connected with an index field for the comparison to work.

**Value**

In the Value field, you enter the comparison value. This value is compared based on the comparison operator in the Condition field with the selected index field.

Depending upon the result of the comparison operation, the document is passed forward to one or the other successor node.

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

![Fig. 46: Decision node in the workflow diagram](image)

The first comparison result determines the path if the comparison is TRUE. This connection is displayed in green.

The second connection is in red. It is used as the path if the result of the comparison is FALSE.

**Distribution node**

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

When you enter a value to 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 provides the query `ELO_WF_STATUS`, which represents a fixed property that is available for all default workflows. Also see *cycle nodes*.

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 passed forward until this occurs.

The following options are available for the collection node:

**Wait for all predecessors**

If the *Wait for all predecessors* option is enabled, the workflow is only passed forward to the next node when all nodes before the collection node have been completed.
**Number of predecessors**

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

**Deactivate these nodes while passing forward**

Use the field *Deactivate these nodes when passing forward* to determine which nodes should no longer be active once the workflow is passed forward. Activate the respective node IDs.

**Information:** You see the node 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 when a process must be repeated until a certain status has been reached.

For a cycle node, you must create a separate cycle node with the option *Cycle beginning* 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 document has to pass through the cycle again. If the condition is met, the cycle is ended. For the comparison to work, you need to connect an index field with the node from which the corresponding value is read.

**Alternative:** You can also enter the query *ELO_WF_STATUS* to the *Index field* field. Use this query to read the workflow status and to check it against the comparison value entered to 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 identifier of the second server must be entered to the server transfer node. After synchronizing the data to the second server, the workflow can be processed further. When the workflow is completed, the data will be transferred back to the first repository.

**Please note:** This option is only necessary when workflows are replicated, i.e. if you have installed the Replication module and are working with more than one broker.

**Subworkflow node**

Once the workflow reaches a subworkflow node, the set subworkflow is started.

**Select template**

![Select template](image)

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 such that they can only be started as subworkflows. For this, disable the option Workflow may be started manually in the start node of the respective subworkflow.
In the *Workflow overview* dialog box, default workflows can be distinguished from subworkflows in the *Type* column. Default workflows can be recognized by the type *Main*. Subworkflows can be recognized by the type *Sub*.

**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. Nodes that are still open are then ignored.

**Information:** If an end note has not been set, the workflow ends when there are no more open nodes.

**Node settings overview**

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

![Workflow overview dialog box](image)

**Fig. 50: "Workflow overview" dialog box**
<table>
<thead>
<tr>
<th>Setting</th>
<th>Start node</th>
<th>User node</th>
<th>Decision node</th>
<th>Collection node</th>
<th>Distribution node</th>
<th>Cycle beginning</th>
<th>Cycle ende</th>
<th>Server transfer</th>
<th>Subworkflow</th>
<th>Ende node</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select second group</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Action button</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wait for all pred.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Condition/value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Comments</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>User/group</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Name (for forwarding)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Name (internal)</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>End script</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>General escalation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Escalation to</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Escalation B and C</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reset successor nodes</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Form</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Index field</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deactivate node</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Only one successor node</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Priority</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Exclude weekends</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Show after</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Script properties</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Start script</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Status</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Symbol</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transfer to server</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Translation variable</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Link group</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Offset</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Select template</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pass forward…</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Manually start workflow</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Editing and managing templates

Every workflow template can be edited retroactively. 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:** When the *Start workflow in edit mode* option is selected, you can still edit workflow templates when starting the workflow. This option is found under *ribbon > ELO > Configuration > Advanced settings > Workflow.*

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

**Edit mode**

1. Open the workflow designer via *ribbon > ELO > 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. Click the *Edit workflow template* button.

The toolbar for editing the workflow template appears.

**Moving nodes**

Nodes can be moved once you have selected a template and activated 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 on the toolbar.

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

**Information:** Existing connections are retained.
Deleting nodes and connections

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

1. Click *Delete* (eraser) on the toolbar.

An eraser icon appears as the cursor.

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

The selected element is deleted without confirmation.

Information: The start node cannot be deleted.

Renaming a 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.

![Fig. 52: 'Rename template' dialog box](image)

The *Rename template* dialog box appears.

2. Change the name of the template.

3. Click OK.

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

Copying a template

Workflow templates can be copied as follows:

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

![Generate a copy of the template dialog box](image)

The *Generate a copy of the template* dialog box appears.

3. Enter a name for the copy of the template.
4. Click *OK*.

The copied template appears in the *Templates* column.
Default workflow

Fig. 55: 'Versions' button

Once you have selected a workflow template and enabled edit mode, you open the Versions of the workflow template dialog box by clicking the Versions button.

Fig. 56: 'Versions of the workflow template' dialog box

In this dialog box, you have the following options:
**Working version:** You can recognize the working version by the check mark in the check box in the Working version column. When you select the check box of another version, this version becomes the working version.

**Date:** You see when which version was created in the 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:** Click the New version button to save the current state of the workflow template as a new version.

**Delete version:** Click the Delete version button to assign a deletion marker (red highlighting) to the selected versions. The version can be deleted entirely via ribbon > Repository > Delete permanently.

**Restore:** Click the Restore button to remove the deletion marker from the selected version.

---

**Permissions**

![Permissions dialog box](image)

*Fig. 57: 'Permissions' button*

Once you have selected a workflow template and enabled edit mode, open the Permissions dialog box by clicking the Permissions button.
Use the *Edit permissions* dialog box to define the permissions to the workflow template.

**Load template**

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

![Load dialog box](image)

The *Load* dialog box appears.

**Optional:** If required, navigate to the storage location for the desired template.

2. Select the desired template.

3. Click *Load*.

The selected workflow template appears in the workflow designer.

**Saving a template**

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

**Information:** To save the current status of a workflow template in ELO, create a new version or click *Apply* in the workflow designer.

1. Select a template that you want to copy in the *Templates* column.
2. Click 'Save'.

![Save button](image1)

Fig. 61: 'Save' button

The 'Save' dialog box appears.

**Optional**: If required, select another storage location.

3. Enter a name for the workflow template.
4. Click Save.
The workflow template is saved externally.

**PDF output**

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

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

![Fig. 63: 'PDF output' button](image)

2. Click the *PDF output* button.
The PDF output dialog box appears.

3. Select the storage location for the PDF file.

Optional: Change the name of the file if you wish.

4. Click Save.

ELO creates a PDF file from the selected workflow template.

Delete template

Remove a workflow template from the ELO repository as follows:

1. Select the workflow template that you want to delete in the Templates.
2. Click *Delete*.

A confirmation dialog will appear.

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

This chapter contains the following topics:

- Basics on form-based workflows
- The form designer
- Creating new forms
- Creating templates
- Toolbar
- Attributes of tables
- Creating tab groups
- Cell properties
- Global form settings
- Integrating a form into a workflow
- Saving form data
- Validation
- Custom styles
- Structure in the repository
In ELO, forms can be used in different locations:

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

This chapter focuses on the interaction between workflows and forms.

**Form and keywording**

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

All index fields located on a keywording form, the extra text, and "map fields" can be used for storing. Map fields are freely definable fields whose contents are saved to the database. The contents of certain map fields can be seen in the keywording via the Additional information tab in the keywording when the user has the corresponding right to it.

Theoretically, an infinite number of map fields can be generated. The fields save information from dynamically created form fields since the map fields are also created dynamically. However, index 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 keywording form for each form and adjust it to 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 Workflow Forms Server (ELOwf) module.

ELO saves HTML, CSS, and JavaScript information of the forms in TXT documents. These TXT documents are found in the ELO repository in Administration » ELOwf Base » Forms.
ELO provides the ELO form designer for creating and editing forms. Open the ELO form designer in the ELO Java Client via ribbon > ELO > 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 > ELO > Form designer.

**Alternative:** 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**
- **(2) Forms column**
- **(3) Components for column**

Fig. 67: Home page of the ELO form designer
**Toolbar**

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

**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 form of templates. With table templates, you have the option to record data in table form and to save it to the database via special map fields.

**Refresh**: Use this function to refresh the form data.

**Close**: Use this function to exit the ELO form designer.

**Assignment**: Use this function to open a submenu. In this submenu, you can assign forms to individual keywording forms. These forms are displayed instead of the keywording preview if set accordingly. The keywording can be edited via the form. If a form has not been assigned, ELO uses a default form.

*Please note*: The Assignment function is not supported in the ELO Windows Client. You can assign forms in the ELO form designer but this will not be visible in the ELO Windows Client.

**Forms**

You see all forms added in the ELO repository in the Forms column. Click a form to select it.

**Components for**

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.
Creating new forms

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

1. Open the form designer.

   ![Form designer](image)

   **Fig. 68: 'Form' button**

2. Click the *Form* button.

   ![Create new form dialog box](image)

   **Fig. 69: 'Create new form' dialog box**

   The *Create new form* dialog box opens.
Optional: To create a form from a keywording form, select a keywording form from the Keywording 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 special characters
- A minimum of one character
- Maximum 21 characters

Information: Due to technical reasons, ELO converts upper case letters to lower case.

4. Click 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 the ELO repository. 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. In the following, the different components of a form are explained.

The advantage of components is that a form can be divided into multiple parts. This way, you have control over which parts of the form are displayed for which node and which parts can be filled in respectively.
The following describes the different tools of the toolbar as they appear when creating templates and tables. The explanations for the toolbar for tab groups can be found in the Creating a tab group section.

'Table' group

![Fig. 70: 'Table' group](image)

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 row**: Use the New row button to add a new row to the grid. The new row appears below the currently selected row.

**Delete row**: Use the Delete row button to delete the currently selected row.

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

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

**Merge**: Use the Merge button to connect the selected cell with the cell that is to the right of it.

**Split**: Use the Split button to disconnect two cells.
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 the Input button to create an input field in the currently selected cell.

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

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 row 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 index 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) Assigning check boxes to multiple cells: When 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 index field (or map field) to each check box field.

![Index field with values from multiple check boxes](image)

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 to the index field (or map field) linked to the check box field when the user clicks a check box in the form. When multiple check boxes are selected, ELO enters the corresponding value to the corresponding field 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 desired image file to the ELO repository under Administration ¶ ELOwf Base ¶ Images.
2. Enter a short name.
3. Open the form designer.
4. Click Refresh.

**Information**: You need to enter the short name to the form later so that the form designer can find the image. (See step 9.)
5. Open the desired form.
6. Open the desired template.
7. Click the desired cell.
8. Click the image button.

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 to 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 the JSAddLine button to create a button in the currently selected cell that the JS_ADDLINE variable has already been allocated to.
Buttons with the variable `JS_ADDLINE` give users the option to duplicate the row above the button when filling out the form.

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

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

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

You can find additional information on validation in the section on Validation.

**Radio button:** Click Radio button to create a radio button field in the currently selected row. 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 index 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) **Distributing radio buttons to multiple cells:** When 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 index field (or map field).

Depending on which option the user selects, the respective value is saved to the corresponding index 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.
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:

\[ \text{IX\_BLOB\_WO\_<NAME>} \]

- **BLOB**: Binary Large OBject
- **WO**: write once

![Fig. 80: Drawing area over a form](image)

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

You can draw a signature in this area. 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.

![Fig. 81: Saved signature](image)

Click OK to save the signature.
**Information**: The signature will not be linked to the form until you have saved the form or passed the workflow forward.

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 allocated to. A button with the variable `JS_REMOVELINE` is used to delete duplicated rows. This deletes the row in which the button is located.

![Fig. 82: Button for deleting a duplicated row](image)

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

**Information**: To delete multiple rows, enter the parameter `lines:` and the number of rows you need in the `Validation` field. To delete the three rows 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 to an empty cell, the cell automatically turns into a text field.

You can enter any text to text fields. For example, as a name for an adjacent field or as information that is permanently displayed.
Text fields also offer the option of storing 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

2. Click the button next to the Text field (in the Properties of the selected cell area).
The *Explanatory text* dialog box appears.

3. Enter the desired text.

4. Click OK.

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

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

Combo box fields provide a list of all terms for selection. The respective selected term is stored to the connected index field (or map field).
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.

Use the Character count field to determine the maximum number of rows that will be displayed in the combo box field. When 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 the Link button 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.

![Fig. 88: Link field](image)

If you left-click the link, the document or website will open in your default browser.

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

The user can enter large text blocks to 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 row that can be linked to your own JavaScript function.

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

JS_<NAME>

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

The *Edit* group contains tools for editing cells.

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

**Cut:** Cut out the contents of the selected cell via the *Cut* button. Use the *Insert* button to insert the copied contents to another cell.

**Information:** The contents of a cell can also be moved to another cell via drag-and-drop actions.

**Copy:** Use the *Copy* button to copy the contents of the selected cell. Use the *Insert* button to insert the copied contents to another cell.

**Insert:** Use the *Insert* button to insert the contents of a previously cut out or copied cell.

'System' group

Fig. 92: The 'System' group
You will find buttons that apply for the entire template in the System group.

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

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

**Save and preview**: Click the Save and preview button to save all changes and open the form in the ELO Browser dialog box. Click the Close button to close the dialog box and go back to the form designer.

**Cancel**: Clicking the Cancel button closes the template without saving the changes. If there are unsaved changes, you need to confirm that you want to discard them.
Creating templates

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

"Standard" templates: These templates – 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 Attributes of tables section.

Tab groups: Tab groups help you 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 Creating a tab group 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.

Creating the basic structure

1. Open the form designer.

2. Select the form that you would like to add a template to.
3. Click Template.

![Create new template dialog box](image)

The Create new template dialog box opens.

4. Use the Name field to enter a name for the new template. The same rules as for naming forms apply. See the Creating new forms section.

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

5. Click OK.
The template is created. The form designer switches into 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) **Form area**: The elements of the template or the table appear here.

(3) **Properties of the selected cell**: Here, you determine the property of the cell selected in the form area.

(4) **Global form settings**: Here, you determine the settings that apply for the entire form.

(5) **Current cell contents**: Shows the HTML structure for the cell selected in the form area.

**Connecting to a key-wording form**

Each form must be connected to a key-wording form. When you create and integrate the key-wording form depends on your preferred way of working.
Connect a form to a keywording 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 keywording form, perform the following steps to connect it to the form:

**Information**: Newly created keywording forms do not appear immediately in the form designer. If required, restart the *ELO Indexserver* and then the *ELO Workflow Forms Services*.

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

![Global form settings](image)

Fig. 96: ‘Keywording form’ field with default settings

You see the current keywording form in the *Global form settings* area in the *Keywording form* field. The field can be edited even when it is grayed out.

4. Click the triangle icon to the right of the *Keywording form* field.
A drop-down menu with available keywording forms appears.

5. Select the desired keywording form.

The selected keywording form is entered.

Creating a 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 the Toolbar section.

Save

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

1. Click Save.

The form designer closes. The added 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, use the Apply or the Save and preview button.
Attributes of tables

Tables are a special form of templates. Table templates consist of dynamically generated tables that users can enter information to. 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 distinct things that you need to note in terms of their functionality and design.

Fig. 98: Basic structure of a table template

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

**Table header**  
The first row 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 rows 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 row**  
Each table only requires one data row. All additional rows are added dynamically when filling out the form. Once the user fills out the first row, another row 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

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

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 row is assigned a number. The column contents are separated by a pipe symbol in the keywording.
Please note: The map name must not contain special characters or spaces.
Creating tab groups

Tab groups are a special form of templates. 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.

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. Click the Tab group button.

Fig. 101: ‘Tab group’ button

Tab group
The Create new template dialog box opens.

4. Use the Name field to enter a name for the tab group.
   The same rules as for naming forms apply (See the Creating a form section).
   **Optional**: Use the Copy from field to apply the settings of an existing template.

5. Click 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, taking into account the following:
**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 row 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, no field will be automatically selected.

---

**Example 1: Simple tab group**

![Form designer](image)

Fig. 104: Tab group with three tabs
In the example above, the three tabs *Order*, *Approval* and *Completed* were added. The form looks like this:

![Sample form with three tabs](image1)

On the selected *Approval* tab, the *approval* template appears in the top part of the form. The *items* 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 the test form](image2)

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

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

You can find additional information in the *Toolbar* section.

**Variable name**
The *Variable name* field serves various purposes:
- Creating a connection to an index field.
- Setting an action for a button.
- Entering variables for automatically completing fields.

![Properties of the selected cell](image)

*Fig. 110: A form field connected to an index field.*

**Create connection to an index field:** Use the drop-down menu for the *Variable name* field to get a list of available index fields. The selection of index fields depends on the keywording form connected to the form. You can see which keywording form is connected to the form in the *Keywording form* field, which is located under *Global form settings*. You can change the keywording form here, if required.
You also have the option of entering the variable directly to the *Variable name* field. The following pattern applies for this:

**IX_GRP_<GROUP NAME OF THE INDEX FIELD>**

![Fig. 111: A button with the variable 'JS_REMOVELINE'](image)

**Setting an action for a button:** When you use a *Button* field type, enter the name of the function to the *Variable name* field to be called when you click 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 row above the button when filling out the form. A button with the variable *JS_REMOVELINE* is used to delete duplicated rows. This deletes the row in which the button is located.

![Fig. 112: An automatically completed form field](image)
**Variables for automatically completing fields:** There are default variables. These variables can be used to automatically insert contents to fields or can be used by scripts for evaluation purposes.

**Information:** When you enter the variables described here to the *Variable name* field, only the corresponding contents are displayed in the form. When you save the value and/or want to process it further, 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:

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

**Additional variables:**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>ELO_CONNECTUSERNAME</td>
<td>The name of the currently logged on ELO user.</td>
</tr>
<tr>
<td>ELO_CONNECTUSERID</td>
<td>The ID of the currently logged on ELO user.</td>
</tr>
<tr>
<td>ELOAS_PATH</td>
<td>The URL of the ELOas used.</td>
</tr>
<tr>
<td>ELO_SERVICEUSERID</td>
<td>The ID of the currently logged on ELOwf service user.</td>
</tr>
</tbody>
</table>
The following variables refer to the settings that were made for the current user in the ELO user manager:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>ELO_USERPROP1</td>
<td>The Windows user name of the current user, if the user is stored to the ELO user manager.</td>
</tr>
<tr>
<td>ELO_USERPROP2</td>
<td>The e-mail address of the current user, if the user is defined in the user manager.</td>
</tr>
<tr>
<td>ELO_USERPROP3</td>
<td>The contents of the Property 5 field.</td>
</tr>
<tr>
<td>ELO_USERPROP4</td>
<td>The contents of the Action field.</td>
</tr>
<tr>
<td>ELO_USERPROP5</td>
<td>The contents of the Property 1 field.</td>
</tr>
<tr>
<td>ELO_USERPROP6</td>
<td>The contents of the Property 2 field.</td>
</tr>
<tr>
<td>ELO_USERPROP7</td>
<td>The contents of the Property 3 field.</td>
</tr>
<tr>
<td>ELO_USERPROP8</td>
<td>The contents of the Property 4 field.</td>
</tr>
<tr>
<td>ELO_SUPERIOR</td>
<td>The ID of the current user's superior.</td>
</tr>
<tr>
<td>ELO_SUPERIORNAME</td>
<td>The name of the current user's superior.</td>
</tr>
</tbody>
</table>

Additional variables:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>ELO_FLOWID</td>
<td>The ID of the current workflow.</td>
</tr>
<tr>
<td>ELO_FLOWNODE</td>
<td>An ID that is composed of the ID of the current workflow and the ID of the current node. For example 118.1.</td>
</tr>
<tr>
<td>ELO_NODEID</td>
<td>The ID of the current node.</td>
</tr>
<tr>
<td>ELO_TEMPLATE</td>
<td>The name of the form used.</td>
</tr>
<tr>
<td>ELO_OBJID</td>
<td>The ID of the ELO object used.</td>
</tr>
<tr>
<td>ELO_TICKET</td>
<td>The ELOas ticket.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Variable</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>IX_ID</td>
<td>Another option for reading the object ID.</td>
</tr>
<tr>
<td>IX_LOCKED</td>
<td>The user who currently has locked the entry.</td>
</tr>
<tr>
<td>IX_CREATEDATE</td>
<td>The filing date/creation date (including time) of the selected entry in ISO format. For example 20140827151800.</td>
</tr>
<tr>
<td>IX_MASKNO</td>
<td>The number of the keywording form used.</td>
</tr>
<tr>
<td>IX_MASKNAME</td>
<td>The name of the keywording form.</td>
</tr>
</tbody>
</table>
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#.

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

**Keyword**: The keyword list of an index field can be used here. You can also enter keyword lists that do not refer to index 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 to the field using a dynamic keyword list.

- **Script name**: Enter the name of the Indexserver script that contains the dynamic keyword list to the Script name field.
- **Filter**: Enter the variables for the required index fields and map fields to the Filter field. If you require multiple fields, you must separate them using commas.

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

**ELO user names**: Allows the selection of users and/or user 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.

![Image of sample settings for a keyword list]

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 filling out the form field, suggestions automatically appear from the stored keyword list that match the user's entry. 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, the field can only be filled out with entries from the keyword list.

**Existing entries:** If the *Existing entries* option is enabled for an input field, ELO suggests terms that have already been entered to this field.
You can open the keyword list with the F7 key. The keyword list also opens automatically as soon as you enter data to a field. ELO then suggests terms that are related to your entry.

**URL**

You can enter the document GUID (including brackets) and the website URL to the URL field.

**Information:** The function is not available for folders.

If you left-click the link, the document or website will open in your default browser.
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**

Use the drop-down menu of the *Display* field to choose from the predefined design features. Change the layout of the selected cell in this manner.

In addition, the elements of the form can be designed via CSS properties. Use the *Form header scripts* component for this.

![Fig. 117: Drop-down menu of the 'Display' field](image)

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

You can find additional information about this in the section on *Custom styles*. 
### 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.

![Tooltip in the form](image)

### Validation

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

![Validation field](image)

### Validation message

Enter additional information for the user in the **Validation message** field. Enter the criteria that are required for the field to be filled out correctly and for the validation to work.
The user see the text as a validation tooltip when text is entered to the field.

**Formula**

The *Formula* field is suitable 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 in general is: 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.
The formula for the net amount field is then:

\[
\frac{\text{IX_MAP\_GROSS1}}{1 + \frac{\text{IX_MAP\_VAT1}}{100}}
\]

**Shortcut**

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

To do so, enter the letter to be assigned to the respective form field to the 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.

You can find additional information in the Toolbar section.
**Write protection**

All fields that the user can enter data to can be assigned write protection. Select the *Read-only* check box to write-protect a field.

**Example**

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

![Sample 'customer' template](image)

Fig. 123: Sample 'customer' template

1. First, create the template *customer*. 
2. 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.

3. Name the copied template *customer_protection* to ensure that the two templates can be clearly distinguished from one another.
4. Select the *Read-only* setting for all fields of the *customer_protection* template.

---

5. Finally, assign the *customer* template to node A and assign *customer_protection* 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.

**Keywording form**

Use the *Keywording form* field to define which keywording form should be used to store the form data.

*Information:* Newly created keywording forms do not appear immediately in the form designer. If required, restart the *ELO Indexserver* and then the *ELO Workflow Forms Services*.

Select the corresponding keywording 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 needed.

**Map name (tables only)**

Use the *Map name* field to determine under which name the table data should be saved. Saving the data takes place via special map fields. One map field is created per table row. Each of the map fields contains the name entered to *Map name* as well as a sequential number. The contents of the map fields are displayed in the keywording under *Additional information*.

Fig. 128: 'Map name' field
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.

![Languages field](image)

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

Translation variable (prefix)

You require the *Translation variable (prefix)* field if you want to provide translated texts using scripts. To do so, you need to use the `.properties` file extension when saving the corresponding text files to the ELO repository.
The following applies for properties files:

- **Character encoding**: UTF-8
- **Path in the ELO repository**: Administration\|Localization\|system
- **One for each language**: A properties file with the corresponding country code (de, en, fr, ...)

A properties file could contain the following:

```
File    Edit    Format    View    Help

pf.messageDays=Please 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.
Please note: You need to restart the ELOix and ELOwf modules so that ELOwf can load the properties file.

**Limited variable access**

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

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

When the *Limited variable access* option is cleared, the contents of index fields that are ignored by the form can be read out, such as via the browser source text of the form.

```javascript
function JS_UPDATE (text) {
    $vars("X_MAP\_DAY") value = text;
    $msgBox("Text", text, 300);
}

function JS\_DAYS(source) {
    inputBox[elo.locale.store["pf\_headDays"], elo.locale.store["pf\_messageDays"], "300","JS\_UPDATE"];
}
</script>
<style type="text/css">
</style>

Fig. 132: Sample script

Fig. 133: Two versions of an input box from the sample script
**Information:** For dynamic templates (such as when using the JS_ADDLINE variable) and in table templates, the Limited variable access option is not valid.

**Realign columns**

The Realign columns option must be selected in table templates for them to work.
To use forms in workflows, you need to determine in the workflow template which form templates should be used in which node. 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 editing the workflow.

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

**Information:** In the ELO Windows Client, it is not possible to view (partial) forms integrated into 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 were created.

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

The node settings of the selected node can now be edited.
4. Click the Select button (to the right of the Workflow form field).

![Workflow form selection dialog box](image)

Fig. 135: 'Workflow form selection' dialog box

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

5. Click the desired form.
The Components for column with the available templates appears in the right area of the dialog box.

6. Click a template.

**Information**: Multiple templates can be selected for a 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.

**Optional**: Click the template again to undo the selection for a template.

7. Click **OK**.

![Fig. 138: 'Workflow form' field](image)

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

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

If you now start a workflow with the corresponding template, the selected template is shown to the workflow participant.

---

**Changing the order**

![Fig. 139: Changing the order](image)
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.

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

You must use the following pattern:

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

Please note: All names must be lower case.

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,customer,article)]
```
Save form data

The data of an ELO form can be evaluated and processed in various ways. For example, you can use the data recorded to determine what the next step is in the current workflow. In addition, the next user in the workflow can view the saved data so that they know how to proceed with the workflow. Form data in ELO can also be subsequently analyzed and used in other business processes.

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

- Via keywording
- Via IX map fields
- Via WO map fields or BLOB fields
- Via WF map fields

**Via keywording**

Fig. 140: Drop-down menu of the 'Variable name' field
To save the data recorded in the form field using keywording, the form field requires a connection to an index field. To do this, you need to select the *Variable name* field from the drop-down menu.

**Information:** The number of index fields is limited to 50. You need to create the index fields you require in advance. Index fields cannot be created dynamically.

**Map fields**

Map fields are intended as an additional data storage option. Map fields are especially necessary if you want to use dynamically generated forms (lines created with JS_ADDLINE). There are two types of map fields.

**IX map fields**

![Properties of the selected cell](image)

Fig. 141: Link to an IX map field

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 repository entry. IX map fields are an extension of the keywording function. Users with the relevant permissions can view the contents of the IX map fields via the *Additional information* tab in the keywording dialog box.

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><START VALUE>
```

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 row containing the IX map field is duplicated using JS_ADDLINE.
Example

The following IX map fields were used in the example: IX_MAP_ARTICLE1, IX_MAP_NOART1, IX_MAP_PRICE, IX_MAP_PIECE1, and IX_MAP_STOT1. As the row with the IX map fields was duplicated using a JS_ADDLINE button in the form, there are two entries (ARTICLE1 and ARTICLE2 etc.) for every IX map field in the keywording 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

Form-based workflow
WO map fields are intended 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 repository 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:

\[ \text{IX\_BLOB\_WO\_<NAME>} \]

The name must not contain special characters or spaces.

**WF map fields**

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 keywording. This means the data is only connected 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 keywording dialog box. Outside of the form, WF map field data can only be accessed via the Indexserver API.
Validation

Validation is the function of checking entered 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.

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 that this validation rule are assigned to only accept date entries. If the date entered is incorrect, a tooltip error message will appear.

**Date format**

Date fields accept the following date formats:
- YYYYMMDD
- YYYY-MM-DD
- DD.MM.YYYY or
- DD.MM.YY

(Y = year/M = month/D = day)

**Year**
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 parameter **num**. Specifies the number of decimal places for numeric inputs. Data entered to 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
num nk:2 = numbers with two decimal places

Number fields accept commas and periods as separators. However, you should not use a combination of these.

For example, you can use 1,23 or 1.23, but not 1,000.23.

**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.

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 rows that the user is allowed to create.

**Example**

```
um 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 way in which these amounts are displayed depends on the language you have selected in the client.

**Example**

When 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.

You cannot save or forward the form.

**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 to the mandatory field. You can only forward the form when all mandatory fields have been completed.
| **asname** | Available for buttons. If you want a button to call an ELOas function, the name of the ruleset to be run 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 row numbers for copied input fields. If the validation field contains the entry count: auto, the value in the new row 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 rows are copied. The default value is one row. |
| **add<*>** | Available for input fields. When selecting multiple terms from a keyword list, this parameter allows you to append the value in the input field 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 character are placed after the predecessor. |
| **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. |
| **Own validation function** | You can also embed your own validation functions using scripts. Validation functions must be named according to the following convention. |
Own 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.

\[ \text{JS}_\text{VAL}_<\text{Name of the function}> \ (\text{fieldName}, \text{field-Value}, \text{param}) \]

\[ \text{JS}_\text{FILTER}_<\text{Name of the function}> \ (\text{front}, \text{inserted}, \text{tail}, \text{param}) \]
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.

**Example**
The following example shows you how to define your own style class and use this in a form.

**Define style settings**

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 bears the name `orange` and is assigned the background color value `#EF8414`.

Fig. 146: 'Edit form header scripts' component; custom style
You would now already be able to apply this style to the form. 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.

Store class in ELO

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

1. Navigate to the Classes folder (Administration\ELOwf Base) in the ELO repository.
2. Open the Keywording dialog box in the Classes folder.
3. Open 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>

We have used the class orange in the example.

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

6. Open the form designer.
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 would like to apply the style to.
3. Open the template you would like to apply the style to.
4. Click the form field that the style should be assigned 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. Click Save.
Integrating 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 via the ELO repository and load them to forms.

1. Save the CSS files to the ELO repository using the following path.

```
Administration\ELOwf Base\Classes
```

The new style is assigned to the selected field.
2. Open the form designer.

3. To load the CSS files, click *Refresh* in the form designer.

4. Open a form.

5. Click *Edit form header scripts*.

6. Select a CSS file from the *Select CSS* drop-down menu.

7. Click *Add*.

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

8. Click *OK* to save the changes.
Structure in the repository

In the repository, all form data is stored to Administration ¶ ELOwf Base.

Fig. 155: Structure in the repository

'Classes' folder

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

This list can be extended as much as you wish. 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 this location. You can find additional information about this in the section on Custom styles (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 export or import it when transferring 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 click Refresh in the form designer to reload the saved forms.
<table>
<thead>
<tr>
<th>Folder</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>'Forms Custom'</td>
<td>The <em>Forms Custom</em> folder is intended for the ELO Business Solutions. This is where you store (partial) forms of an ELO Business Solution that share the same name. You can then edit and customize 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 <em>Forms Custom</em> folder, ELO always uses the settings for the copies instead of those for the original forms.</td>
</tr>
<tr>
<td>'Images' folder</td>
<td>In the <em>Images</em> folder, you can store your own image files, which are then available through the <em>Image</em> function in the editor.</td>
</tr>
<tr>
<td>'Preview' folder</td>
<td>The <em>Preview</em> folder is used for previewing forms in the form designer. As a form always has to be linked to an ELO object for you to be able to view them, the <em>Preview</em> folder is used as a wildcard object for the preview.</td>
</tr>
<tr>
<td>'Webapp' folder</td>
<td>You can store your own script files or HTML files in the <em>Webapp</em> folder. These files are copied to the <em>Webapp</em> server directory when you start the program or refresh the repository and can be used when you are working in the form.</td>
</tr>
</tbody>
</table>
Advanced functions

This chapter contains the following topics:

- Scripts in the workflow
- End workflow
Using scripts

By using scripts, you can integrate additional functions and automated processes into workflows.

**Information:** Please note that this manual does not provide basic information about scripting in general.

**Types of scripts**

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 using the *Edit form header scripts* component.

- **Script language:** JavaScript
You can save any number of additional script functions here. To prevent the scripts conflicting 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_`.

### Events

The following contains a short description of the default events in the *Edit form header scripts* component.

**onInit():** The event `onInit` is called once when the form is loaded.

**inputChanged(elem):** The event `inputChanged` is called when the user changes an input field and then closes the field. In this case, the parameter `source` contains the respective input field. In addition, this function is called when the form is displayed in order to permit initializations to occur. If this is the case, the parameter `source` is output as `undefined`. 
**nextClicked(id)**: The event `nextClicked` is called after the user has passed forward the workflow but before the forwarding action is actually carried out. The `id` parameter tells you which node the workflow was passed forward from. If required, you can cancel forwarding by entering `false` as the return value.

**saveClicked()**: The event `saveClicked` is executed when the user clicks the Save button but before the save action is actually carried out. If required, you can cancel saving by entering `false` as the return value.

**addLineClicked**: The event `addLineClicked` is executed as soon as the `JS_ADDLINE` function is executed. The `addLineId` represents the parameter ID of the respective row. The `groupIndex` parameter represents the respective index field variable. If required, you can cancel the action by entering `false` as the return value.

**removeLineClicked**: The event `removeLineClicked` is called as soon as the `JS_REMOVELINE` function is executed. The `addLineId` represents the parameter ID of the respective row. The `groupIndex` parameter represents the respective index field variable. If required, you can cancel the action by entering `false` as the return value.

**tabChanged**: The event `tabChanged` is executed when the user switches to another tab in the form. The `id` parameter refers to the tab ID of the respective tab group. If required, you can cancel the action by entering `false` as the return value.

---

**Help functions**

To support user functions, there are a number of helper routines that make it easier to program frequently used actions. These helper routines are listed below and explained based on examples.

**$val(<name>):** Use this function to query the contents of an input field with the name `<name>`. This function returns the name as a text string.

Example

```javascript
var name = $val("IX_GRP_NAME");
```

**$num(<name>):** Like the function `$val`, this function returns the contents of the field with the name `<name>`. In this case, however, it is returned as a numeric value and not as text.

Example

```javascript
var vat = $num("IX_GRP_VAT");
```
$vat(<name>): To set a value, you can also determine the variable which contains the input field. If there are several input fields with the same name, the first input field that is not write-protected is selected.

Example
$var("IX_GRP_VAT").value=19;

$update(<name>,<value>): This function enters the specified value to the <name> input field and then calls 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.

formatDate(<isoDate>): This function converts an ISO date entry into the standard date form DD.MM.YYYY.

Example
alert(formatDate("20110131");

toDay(): This function returns the current date.

addErrorMsg(source,msg): This function shows an error message (msg) for an input field (source).

sumAllInput(sourceItems,destination): This function adds the values of a column together which were created by copying input lines. These columns always have the naming convention <name><counter>, e.g. VALUE1, VALUE2 etc. The column is thus named VALUE (sourceItems).

Example
If you now call sumAllInput("VALUE","SUM"), the system searches for all input fields named VALUE1, VALUE2, etc., adds them together, and writes the result to the input field with the name SUM.

getNumberOfWorkDays(startDate,endDate): This function determines the number of weekdays between two dates. The startDate might be after the endDate; in this case, the order will be reversed automatically. The number of days cannot be negative.

Please note that this function does not take holidays into account. Integrating this additional function would be rather complicated, as in addition to fixed holidays, there are also movable holidays that would need to be calculated. At the same time, these holidays would also depend on the various regulations of each state or national government.
inputBox(): You can use this function for user input in any folder. It can be used in your own scripts by implementing the following call: `inputBox(<Title>,<Message>,<Width>,<Ok function>).`

Example

![Image of an input dialog box](image)

Fig. 159: Example of an input dialog box

```javascript
function JS_DAYS (source) {
  inputBox ("Number of days","Enter the desired number of days:","300","JS_UPDATE");
}

function JS_UPDATE (text) {
  $update("IX_GRP_DAYS",text);
}
```

The `JS_DAYS` function calls the input dialog box. If the user clicks OK, the `JS_UPDATE` function is called. The parameter of this function is the text entered by the user. This is saved to the field containing the variable `IX_GRP_DAYS`.

Please note: Functions to be embedded using buttons require the prefix `JS_` in the name and must be written in capital letters.

msgBox(): This function calls a message dialog box. The following pattern applies for this:

```javascript
msgBox (<Title>, <Message>, <Width>)
```
Advanced functions

**Example**

function JS_SUCCESS () {
    msgBox ("Success","The action was successfully completed.", "300");
}

eloAlert(): This function calls a warning dialog box. The following pattern applies for this:

eloAlert (<Message>, <Title>)

**Example**

function JS_ALERT (){
    eloAlert("Error while executing", "Warning");
}

getInput(source): This function is called when a button is clicked and returns the button which was used for the call. In Firefox, the button is retrieved from the source event, whereas in Internet Explorer, the button is retrieved from `window.event`.

hidelInputs("<Field variable1>", "<Field variable2>", ...): This function is used for hiding the form fields that are transferred as parameters.
showInputs("<Field variable1>", "<Field variable2>", ...): This function is used for showing the form fields that are transferred as parameters.

hideCells("<CSS class1>", "<CSS class2>", ...): This form is used for hiding one or multiple cells with the same formatting (CSS class).

hideCells("<CSS class1>", "<CSS class2>", ...): This form is used for showing one or multiple cells with the same formatting (CSS class).

**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 Indexserver.

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

There are two storage locations for the scripts in the ELO repository:

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

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

**Information**: Newly entered 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.

For the ELO Java Client to recognize start scripts, they must contain the following functions:

- **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 user
- **workflow**: The current workflow
- **nodeId**: ID of the respective node

**End scripts**
Use end scripts to define actions that will be run when passing forward 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**

Use the *Action buttons* field (in user nodes) to integrate up to five action scripts into a workflow node. Click the *Select* button (button to the right of the *Action buttons* field) to open the *Action scripts* dialog box. Select the desired scripts.

**Information**: If you want to use action buttons, you have to enter at least two action scripts.
Action buttons for the respective node appear as additional buttons in the Pass workflow forward 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 the ELO repository.

The action script should include the following functions. Replace the wildcards (such as [NAME]):

```javascript
function cfb<NAME>Start(){
}

function cfb<NAME>Name(){
    return "<LABEL>";
}

function cfb<NAME>Tooltip(){
    return "<TOOLTIP>";
}
```
Once the script has been saved at the above-mentioned location in the ELO repository, you need to reload the scripts. Use the keyboard shortcut CTRL + ALT + R for this.

Example

The following example illustrates what an action script for an action button could look like. The action stored here opens an empty Microsoft Excel document. The required Jacob classes for this (Jacob = JavaCOM Bridge) are imported via the first lines of the script.

```javascript
//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);
}

//Label of the button
function cfbOpenExcelName(){
  return "New Excel document";
}

//Tooltip for button
function cfbOpenExcelTooltip(){
  return "Open a new document in Microsoft Excel";
}
```
End workflows

Workflows are usually completed when the last node finishes executing. 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, this *cannot* be reversed.

1. Click *Workflow overview (ribbon > Tasks)*
   
The *Workflow overview* dialog box appears.
2. Select the workflow you want to terminate.

![Fig. 164: 'End workflow' menu items](image)

3. Open the context menu.
4. Click *End workflow*.
The *End workflow* dialog box appears.

5. Confirm the dialog box with *Yes*.

The selected workflow will be terminated.

In the *Workflow overview* dialog box, you can set the *Finished* 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.
Fig. 166: Additional node in a workflow that was terminated manually

In addition, the date of termination is recorded in the user node.
Appendix

Disclaimer of Liability

ELO Digital Office GmbH, Tübinger Strasse 43, 70178 Stuttgart, Germany
ELO Digital Office AT GmbH, Leonfeldner Strasse 2-4, 4040 Linz, Austria
ELO Digital Office CH AG, Lagerstrasse 14, 8600 Dübendorf, Switzerland
The ELO homepage is available at: www.elo.com
ELO Digital Office, the ELO logo, elo.com, ELOoffice, ELOprofessional and ELOenterprise are trademarks of ELO Digital Office GmbH in Germany and/or other countries. Microsoft®, MS®, Windows®, Word® and Excel®, PowerPoint®, SharePoint®, Navision® are registered trademarks of Microsoft Corporation in the USA and/or other countries. Other company, product, or service names may be trademarks of other companies.
This publication serves only as non-binding general information and is not a substitute for a detailed, individual consultation. ELO reserves the right to change information in this publication at any time without prior notification. Technical characteristics and functions may vary, particularly from country to country.
You can obtain the latest information to ELO products, contractual conditions and prices from the ELO associations, ELO Business Partners, and/or from ELO Channel Partners. The product information reflects the present status. The subject and extent of the services are exclusively defined in the corresponding contracts.
ELO does not guarantee that its products or other services ensure compliance with specific legislation. The customer is responsible for the adherence to security regulations and other standards of national and international law.
We reserve the right to make changes and assume no liability for errors and misprints. Reproduction and/or distribution, in part or in whole, is only allowed with written consent from ELO Digital Office GmbH.
© Copyright ELO Digital Office GmbH 2017. All rights reserved.
Index

Action buttons - 44, 154
Action scripts - 154
Add, component to tab group - 99
Add, form - 71
Add, nodes - 34
Add, participants - 15
Add, script - 147
Add, style - 136
Add, tab group - 97
Add, template - 89
Add, workflow template - 27
Additional information - 67
Ad hoc workflow - 10, 13
Ad hoc workflow, add participants - 15
Ad hoc workflow, approval - 15
Ad hoc workflow, create - 13
Ad hoc workflow, end script - 19
Ad hoc workflow, escalation management - 18
Ad hoc workflow, inform/distribute - 15
Ad hoc workflow, job instructions - 17
Ad hoc workflow, maximum duration - 18
Ad hoc workflow, name - 14
Ad hoc workflow, not approved notification - 18
Ad hoc workflow, recipient - 18
Ad hoc workflow, success notification - 18
Administration folder - 68
AND group - 38
Apply, template - 88
Approval, ad hoc workflow - 15
Autofill - 108
BLOB - 80, 130
Button - 104
Cancel, template - 88
Cell - 87
Cell contents - 76, 91
Cell properties - 103
Character count - 74, 113
Character encoding - 119, 153, 155
Check box - 74, 74
Classes - 137, 142
Close, form designer - 70
Collection node - 24, 48
Combo box - 83
Comment, version - 58
Comments - 40
Components, form designer - 70
Condition - 46, 49
Connect, node - 23, 35
Connection, delete - 24, 55
Conventions - 7
Copy, cell contents - 87
CSS - 67, 110, 136, 142
Current cell contents - 76, 91
Cut - 87
Cycle node - 25, 49
Data, save - 127
Date, version - 58
Date field - 78
Date field, keyword list - 107
Decision node - 24, 46
Default workflow - 10
Delay - 42
Delete, cell contents - 87
Delete, component on tab group - 99
Delete, connection - 24, 55
Delete, node - 24, 55
Delete, version - 58
Delete, workflow template - 63
Delete column - 73
Delete function - 24
Delete row - 73
Display - 110
Distribution node - 24, 47
Drag-and-drop, cell contents - 87
Due date - 42
Dynamic form - 121
Dynamic keyword list - 107
Edit, node settings - 37
Edit mode - 29, 54
Editor field - 85
ELO_wf_STATUS - 48, 49
ELOas - 48, 49, 107
ELO form designer - 69
ELO user names - 107
ELOwf - 67
End node - 25, 51
End script - 19, 41, 154
Eraser - 24, 55
Escalation management - 18, 31
Exclude weekends - 31
Existing entries - 108
Explanatory text - 82
Field, complete automatically - 105
Field type - 74, 103
Form - 34, 43, 67, 70
Form, autofill - 108
Form, BLOB - 80, 130
Form, cell - 87
Form, cell properties - 103
Form, character count - 74, 113
Form, check box - 74, 74
Form, combo box - 83
Form, create - 71
Form, CSS - 110
Form, date field - 78
Form, delete column - 73
Form, delete row - 73
Form, display - 110
Form, drag-and-drop - 87
Form, drop-down menu - 84
Form, dynamic - 121
Form, editor field - 85
Form, existing entries - 108
Form, field type - 74, 103
Form, form columns - 116
Form, form header scripts - 104, 110, 136, 147
Form, formula - 112
Form, global settings - 92, 117
Form, GUID - 85, 109
Form, help functions - 149
Form, image field - 76
Form, index field - 104
Form, input field - 74
Form, integrate in workflow - 122
Form, IX map - 128
Form, JavaScript - 86, 147
Form, JS_ADDLINE - 77
Form, JS_REMOVELINE - 81
Form, keyboard shortcut - 113
Form, keywording form - 67, 91, 117
Form, keyword list - 107
Form, languages - 118
Form, layout - 110
Form, limited variable access - 120
Form, link - 84, 109
Form, map fields - 67, 70, 89, 94, 128
Form, map name - 95, 117
Form, merge cells - 73
Form, name - 72
Form, new column - 73
Form, new row - 73
Form, only list values - 108
Form, properties - 118
Form, properties of the cell - 103
Form, radio button - 78
Form, realign columns - 121
Form, save - 93
Form, save data - 127
Form, selection field - 78
Form, selection list - 84
Form, shortcut - 113
Form, signature field - 79
Form, split cells - 73
Form, tab group - 89, 97
Form, table - 73, 89, 94
Form, template - 89
Form, text - 103
Form, text field - 81
Form, tooltip - 111
Form, translation - 118
Form, URL - 85, 109
Form, validation - 111, 131
Form, variable - 103
Form, WF map - 130
Form, WO map - 80, 130
Form, write protection - 114
Form area - 91
Form columns - 116
Form designer - 69
Form designer, components - 70
Form designer, configuration - 142
Form designer, open - 26
Form designer, structure in the repository - 142
Form designer, style names - 143
Form designer, toolbar - 70
Form header scripts - 104, 110, 136, 147
Formula - 112
Function, name - 148
Global form settings - 91, 92, 117
Group nodes, link - 38
GUID - 85, 109
Header, tab group - 99
HTML - 67, 145
Icon, node - 29
ID, node - 49
Image field - 76
Images - 145
Import, form - 144
Import, workflow template - 60
Index field, value to be checked - 47, 49
Index field, variable - 104
Index fields - 46
Indexserver, scripts - 153
Inform/distribute, ad hoc workflow - 15
Input field - 74
Input field, keyword list - 107
Insert, cell contents - 87
Internal name - 31
IX map - 128
JavaScript - 67, 86, 147, 153, 155
Job instructions - 17
JS_ADDLINE - 77
JS_REMOVELINE - 81
Keyword - 107
Keywording - 67, 91
Keywording form - 67, 117
Keyword list - 107
Languages - 118
Layout - 110
Limited variable access - 120
Link - 84, 109
Link, group nodes - 38
Load, template - 60
Map fields - 67, 70, 89, 94, 128
Map name - 95, 117
Maximum duration - 18, 32
Merge - 73
Minimized view - 26
Move - 54
Name, ad hoc workflow - 14, 14
Name, form - 72
Name, function - 148
Name, tab group - 98, 98
Name, template - 90, 90, 117
Name, workflow template - 28, 28, 55, 55
Name for forwarding - 41
New column - 73
New row - 73
New version - 58
Node, cycle node - 25, 49
Node, deactivate - 49
Node, decision node - 24, 46
Node, delete - 24, 55
Node, distribution node - 24, 47
Node, end node - 25, 51
Node, server transfer - 25, 50
Node, subworkflow - 25, 50
Node, user node - 24, 35, 37
Node ID - 49
Node properties, second group - 38
Nodes, add - 34
Nodes, collection node - 24, 48
Nodes, connect - 23, 35
Nodes, move - 23, 54
Nodes, select - 23
Nodes, timeout - 24
Node settings - 22, 37
Node settings, action buttons - 44
Node settings, comments - 40
Node settings, condition - 46, 49
Node settings, deactivate node - 49
Node settings, delay - 42
Node settings, due date - 42
Node settings, end script - 41
Node settings, escalation management - 31
Node settings, exclude weekends - 31
Node settings, form - 34, 43
Node settings, icon - 29
Node settings, index fields - 46
Node settings, internal name - 31
Node settings, link group - 38
Node settings, manually start workflow - 33
Node settings, maximum duration - 32
Node settings, name for forwarding - 41
Node settings, owner - 30
Node settings, passed deadline - 33
Node settings, pass forward at the number of completed predecessor nodes - 49
Node settings, priority - 30
Node settings, script properties - 34
Node settings, show after - 42
Node settings, spacing - 49
Node settings, start script - 33
Node settings, status - 48
Node settings, transfer to server - 29
Node settings, translation variable - 31, 41
Node settings, user - 30, 37
Node settings, value - 47, 49
Node settings, wait for all predecessors - 48
Not approved notification - 18
onEnterNode - 153
onExitNode - 154
Only list values allowed - 108
Owner - 30
Participants, ad hoc workflow - 15
Passed deadline - 33
Pass forward at the number of completed predecessor nodes - 49
PDF output - 62
Permissions - 58
Preview, form designer - 70
Priority - 30
Properties, cell - 91, 103
Properties file - 118
Radio button - 78
Read mode - 29
Realign columns - 121
Recipient, ad hoc workflow - 18
Refresh, form designer - 70
Rename, template - 55
Replication - 29, 50
Restore, version - 58
Save, form data - 127
Save, preview - 88
Save, template - 61, 88, 93
Scripts - 19, 33, 41, 44, 48, 49, 76, 104, 119, 145, 147
Scripts, validation - 134
Second group, selecting - 38
Select, second group - 38
Select, user - 37
Selection field - 78
Select tool - 23
Server transfer - 25, 50
Settings, form - 91, 117
Settings, nodes - 22, 29, 37
Settings, validation - 131
Settings, workflow - 22, 29
Shortcut - 113, 113
Show after - 42
Signature field - 79
Spacing - 49
Split - 73
Start element - 99
Start node, form - 122
Start script - 33
Start workflow in edit mode - 54
Status - 48
Style - 136, 142
Subworkflow - 10, 25, 33, 50
Success notification - 18
Tab group - 70, 89, 97
Tab ID - 99
Table - 70, 73, 89, 94
Table, map name - 95, 117
Target audience - 8
Template - 70, 73
Template, create - 89
Template, creating - 27
Template, name - 90, 117
Template, save - 88, 93
Template, show - 88
Template, version - 57
Text field - 81, 103
Timeout, successor nodes - 24
Toolbar - 23, 91
Toolbar, form designer - 70
Tools - 23
Tooltip - 111
Transfer to server - 29
Translation - 118
Translation variable - 31, 41
TXT - 68
Type, workflow - 51
URL - 85, 109
Use grid - 26
User - 30, 37
User, pass forward - 42
User node - 24, 35, 37
User nodes, form - 122
UTF-8 - 119, 153, 155
Validation - 111
Value - 47, 49
Variable - 103
Version, workflow template - 57
Wait for all predecessors - 48
WF map - 130
WO map - 80, 130
Workflow, configuration - 54
Workflow, connect with form - 122
Workflow, edit - 54
Workflow, integrating scripts - 147
Workflow, pass forward - 154
Workflow, replication - 29
Workflow, start manually - 33
Workflow, type - 51
Workflow, types - 10
Workflow designer - 21, 23
Workflow form - 34
Workflow forms - 26, 43, 67, 122
Workflow settings - 22, 29
Workflow status - 48, 49
Workflow template - 27, 54
Working version, workflow template - 58
Write version, workflow template - 58
Write protection, form field - 114