On Alfresco ECM Platform

Case Management

2016

1. Case Creation and Setting 3
2. Roles 7
3. Activities 9
  3.1 Action 9
  3.2 Stage 14
  3.3 Task 17
4. Events 23
5. Event Conditions 34
6. Business Requirements 39
7. Case Template 52
8. Case Import and Export 54
  8.1 Export 54
  8.2 Import 59
9. Advanced Topics 62
  9.1 Extending the list of available tasks in a case 62
  9.2 Dynamic roles 73

1. Case Creation and Setting

From the point of view of the system, a case is an object with the aspect **icase:case** or "Is a case".

To illustrate it, we will go to the repository and create a conventional folder named "CaseTest" in the guest catalogue.

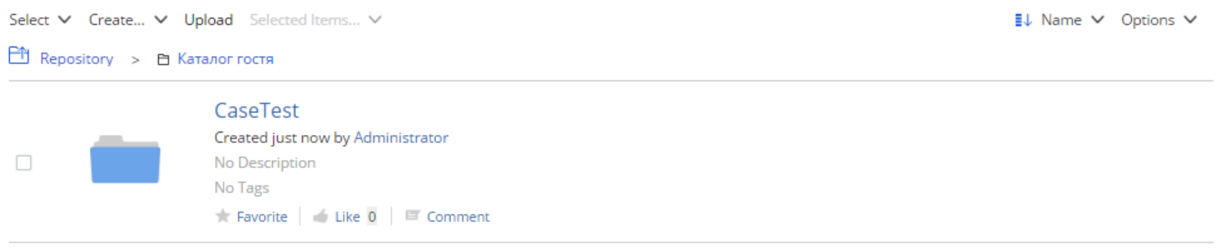


Figure 1 - Just a folder

On the right, we open an additional list of actions (More...) and select "Aspect Setting" as shown in Figure 2.

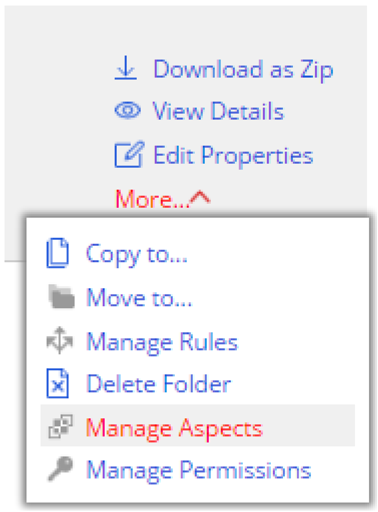


Figure 2 - Aspect setting selection

In the opening window, we have to select the "Is a case" aspect.

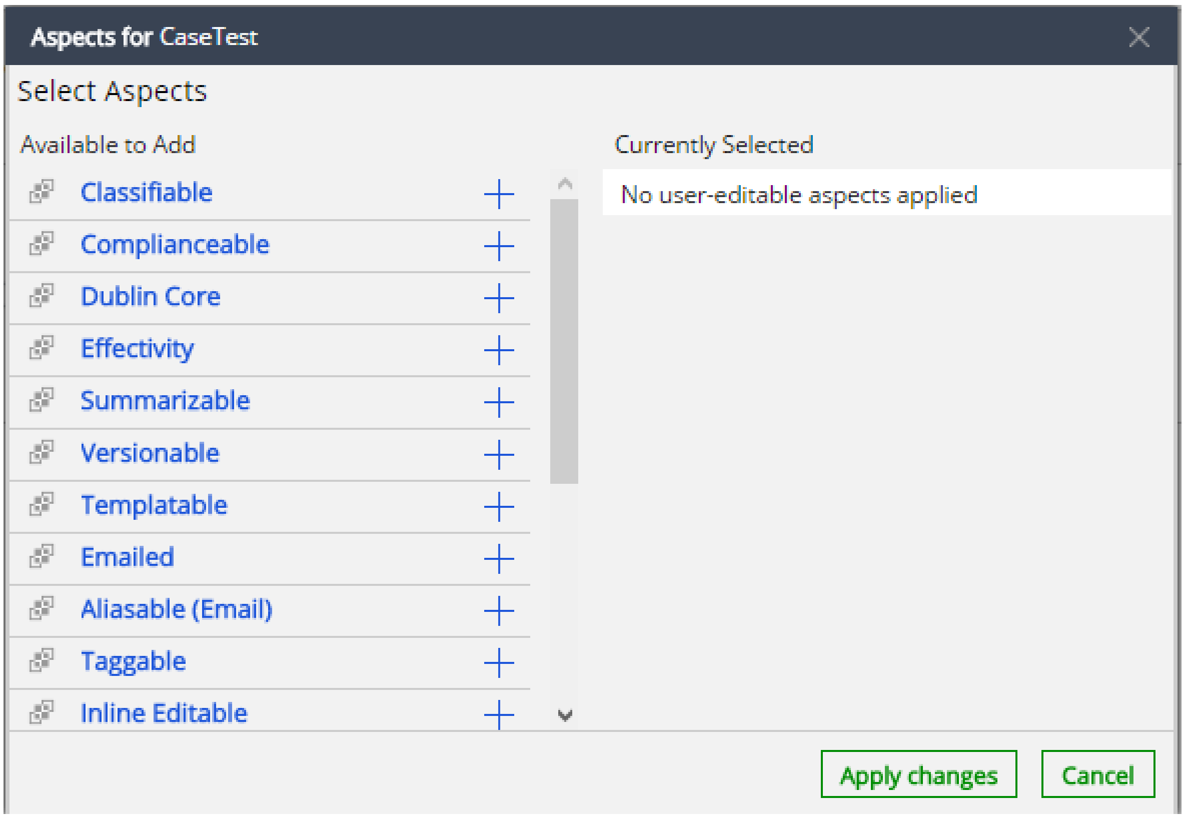


Figure 3 - Aspect setting

We click "Apply Changes" and we are routed to the card of the created folder. This can be done by clicking "Details" on the actions list.

In the case card, we will see a "Case Setting" tab. (see Figure 4).

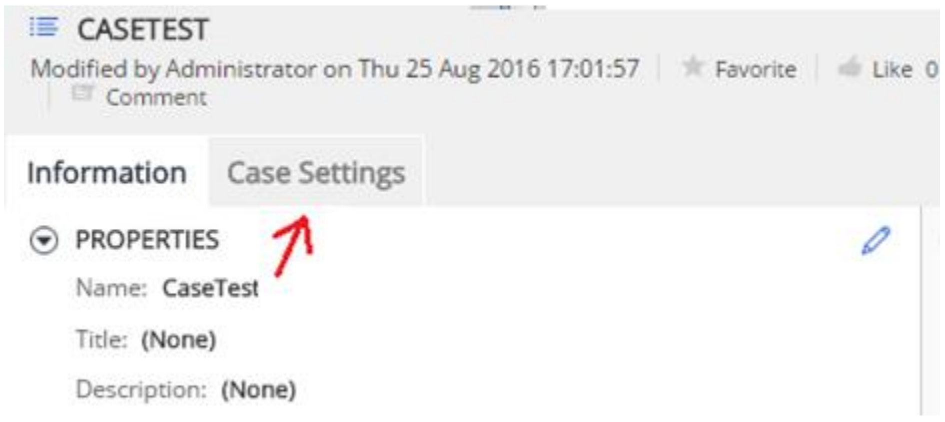


Figure 4 - "Case Setting" tab

Now we go to case setting and add new types of case elements. For this, we need to click the button pointed with an arrow in Figure 5.

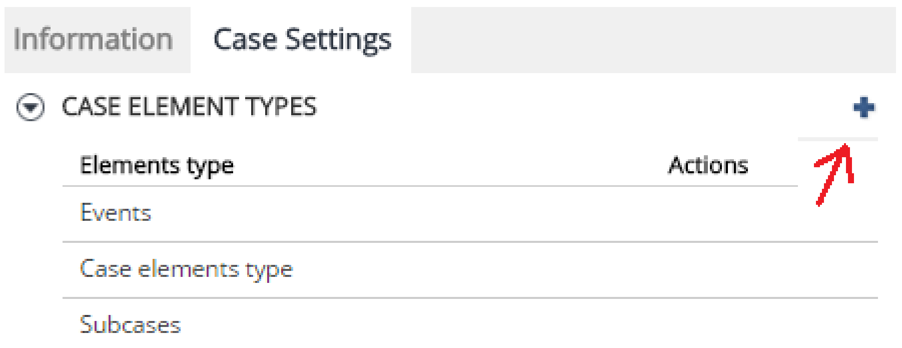


Figure 5 - "Add New Case Elements" button

After clicking that button, a window will open enabling selection of case elements. If the list is empty, in order to call the selection list we need to enter the asterisk "*" symbol in the search field and press Enter. On the appearing list we select "Tasks", "Roles" and "Business Requirements". After these actions, the window should look as shown in Figure 6.

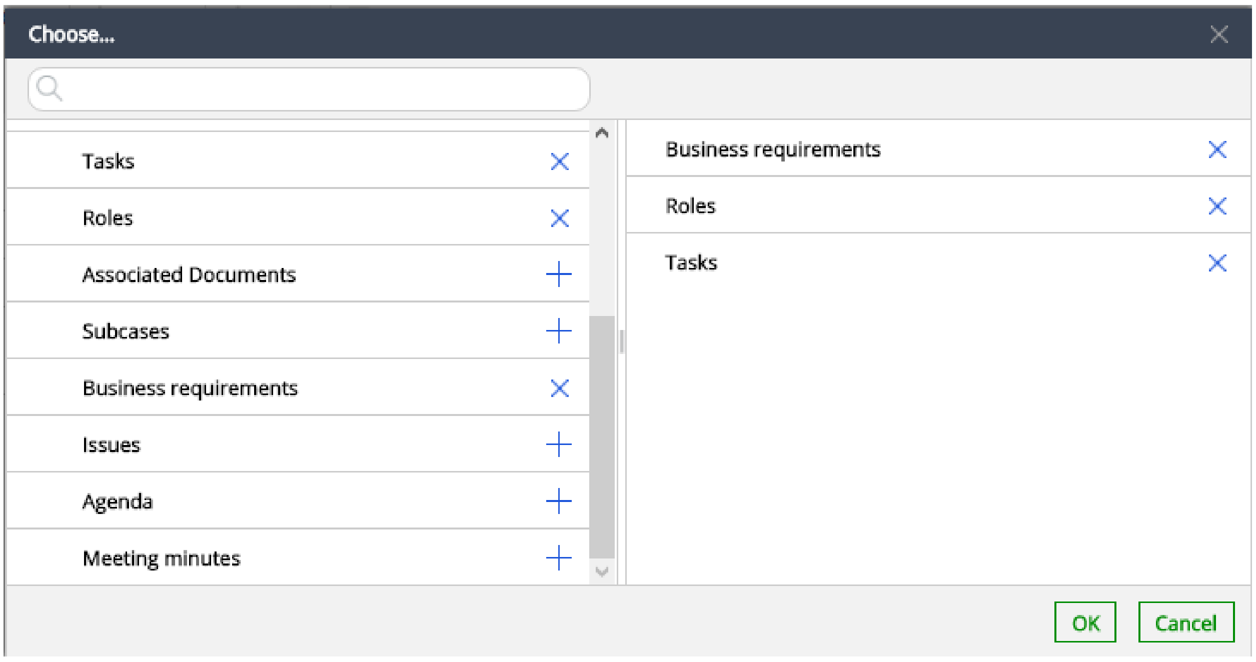
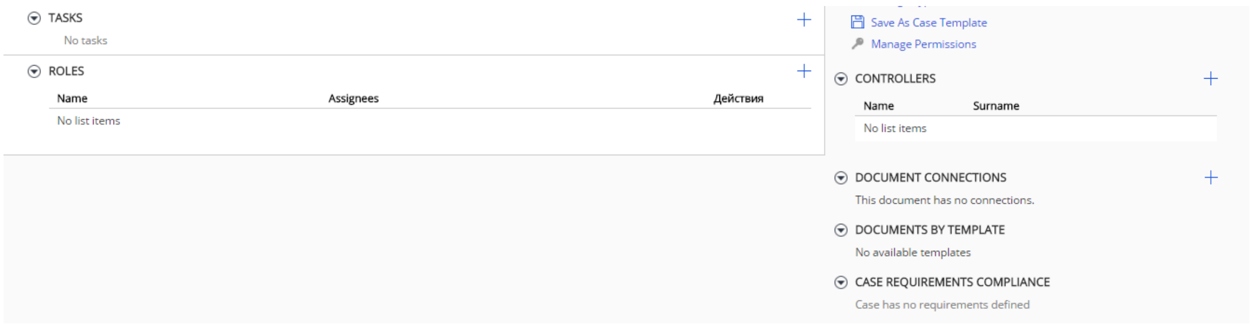


Figure 6 - Case elements selection window

We click "OK" and return to the "Card" tab. If we did everything correct, three new cardlets will appear on the card: "Tasks", "Roles" and "Case Compliance" (Figure 7).



TASKS

No tasks

ROLES

Name	Assignees	Действия
No list items		

Save As Case Template

Manage Permissions

CONTROLLERS

Name	Surname
No list items	

DOCUMENT CONNECTIONS

This document has no connections.

DOCUMENTS BY TEMPLATE

No available templates

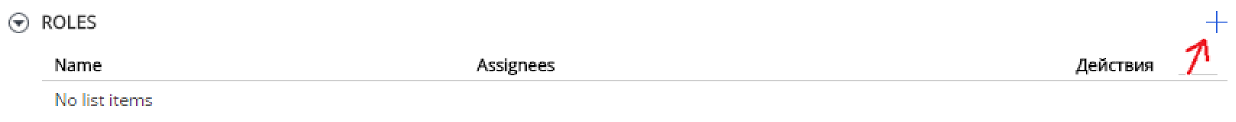
CASE REQUIREMENTS COMPLIANCE

Case has no requirements defined

Figure 7 - New cardlets

2. Roles

Roles are designed for selecting people or groups related to a case. In the activity of a "Task" case there is always one or several roles participating. We will create a new role by clicking the button marked in Figure 8.


ROLES

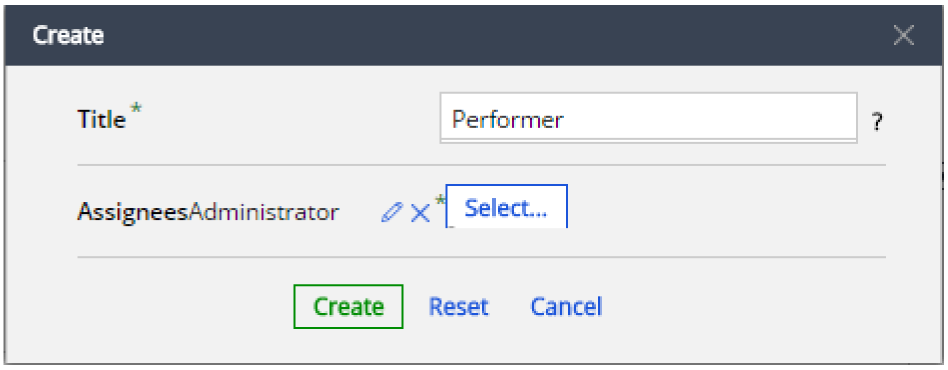
Name	Assignees	Действия
No list items		

Figure 8 - Button for new role creation

In the appearing form, we can see two fields:

- Title - Role name. Can be arbitrary;
- Representatives: Individuals or groups representing this particular role. Representatives are selected from the orgchart.

After the fields are completed, the form should look as shown in Figure 9.

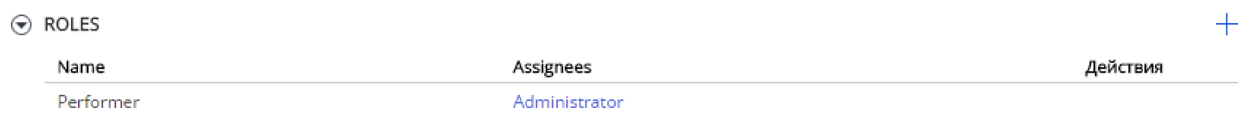

Create

Title * : Performer

Assignees: Administrator * Select...

Figure 9 - Completed form for new role creation

We click on "Create" and can see a new role appearing in the "Roles" cardlet (Figure 10). Thus, one can create an unlimited number of roles that can be assigned with tasks.



ROLES

Name	Assignees	Действия
Performer	Administrator	

Figure 10 - Added role

Role representatives can be changed at any time, but one should keep in mind that when a case task is started, all representatives are retrieved from the role and are assigned BPMN tasks. Thus, if the representatives are changed after the task start, this will have no influence on the already started task.

3. Activities

By pressing the blue + in the "Tasks" cardlet we can select one of the three activity types (see Figure 11).

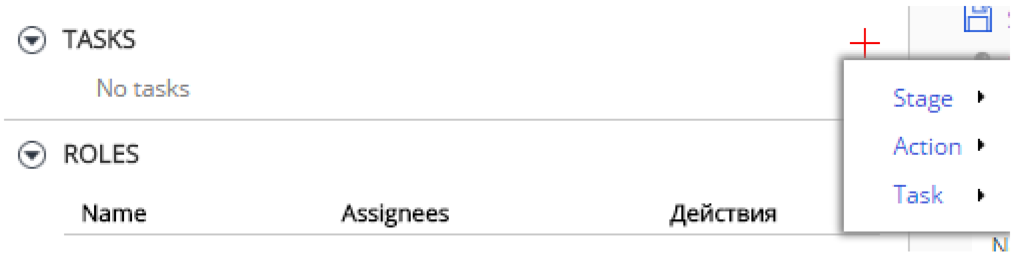


Figure 11 - New activity creation

3.1 Action

Action is an automatic activity, which starts and ends within the course of one transaction. It is used in cases when human interference is not required.

Reaction to an event: action start (start event)

Types of actions:

- **Send e-mail**

  Send an e-mail with a specified subject and text to a specified recipient.

  Properties: Subject, Recipient, Text

- **Set document property**

  Record a specified value into a specified property of the current document (case). Example of a property: **cm:title**.

  Properties: Property, Value

- **Specify process variable**

  Specify a process variable.

  Properties: Variable, Value

- **Start javascript**

  Execute an arbitrary javascript. The context allows access to the following objects: **document**: current document (case), **event**: event that triggered action start.

  Properties: Title, javascript-code

- **Show error message**

  Upon start of this activity exception is thrown. The user is shown an error message and the transaction is rolled back.

  Properties: Error message

Example:

We will create a sample action "set document property" (Figure 12).

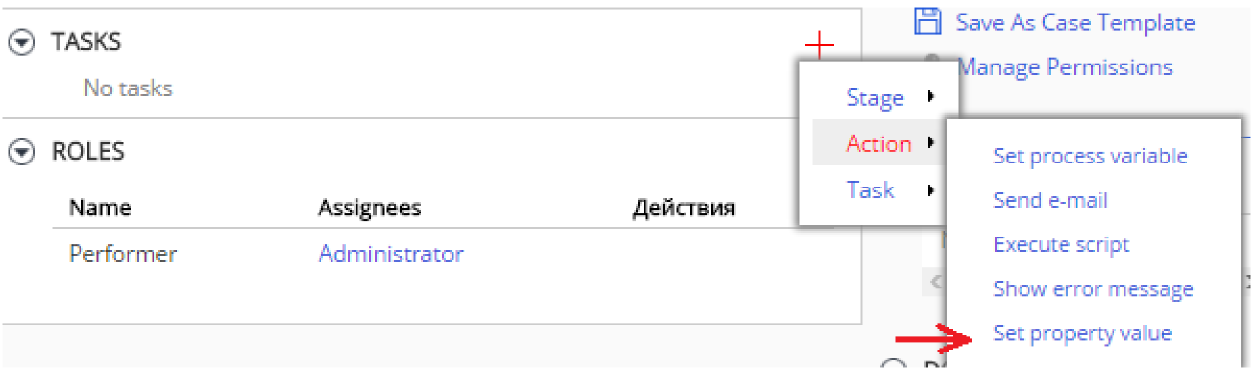
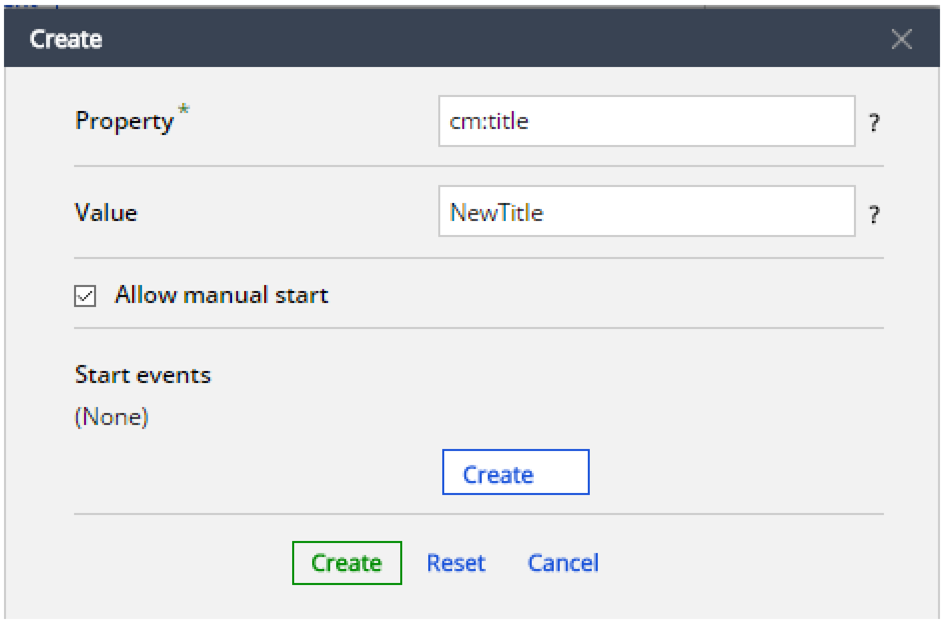


Figure 12 - Creating a "Set Document Property" action

We fill in the form as shown in Figure 13. (Property: **cm:title**, Value: **NewTitle**).



Property *	cm:title
Value	NewTitle

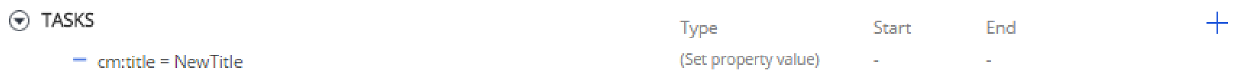
☑ Allow manual start

Start events

(None)

Figure 13 - Filling the Create Action form

We click the green "Create" button at the bottom of the form and we can see that our action has appeared in the "Tasks" cardlet (Figure 14).

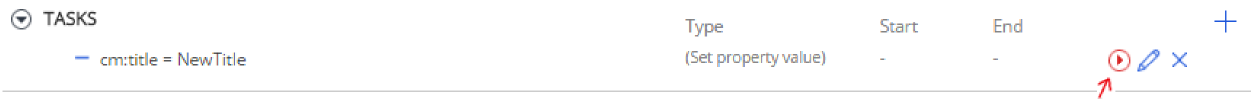


TASKS

	Type	Start	End
cm:title = NewTitle	(Set property value)	-	-

Figure 14 - New action

We start the action as shown in Figure 15.

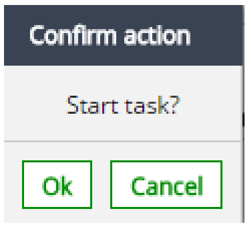


TASKS

	Type	Start	End
cm:title = NewTitle	(Set property value)	-	-

Figure 15 - Action start

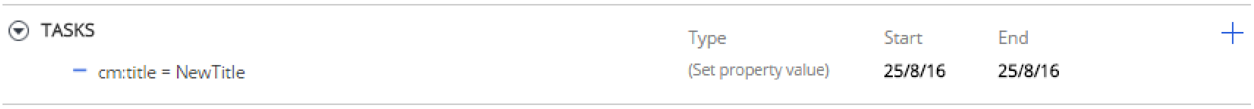
When the system asks, "Start Task?" we click "Yes" (Figure 16).



Start task?

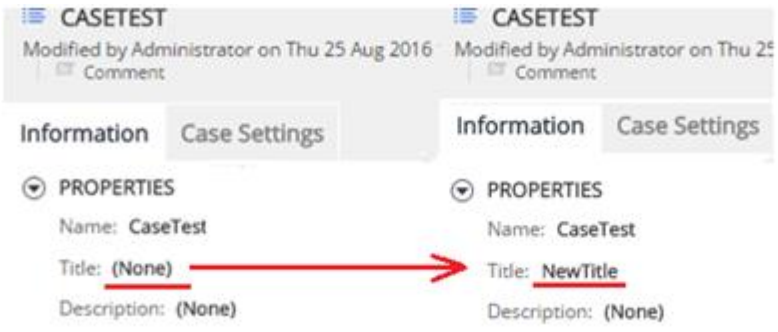
Figure 16 - Confirmation of task start

After the confirmation, we can see the following results:



	Type	Start	End
cm:title = NewTitle	(Set property value)	25/8/16	25/8/16

Figure 17 - Activity start and end dates have appeared



Modified by Administrator on Thu 25 Aug 2016
Comment

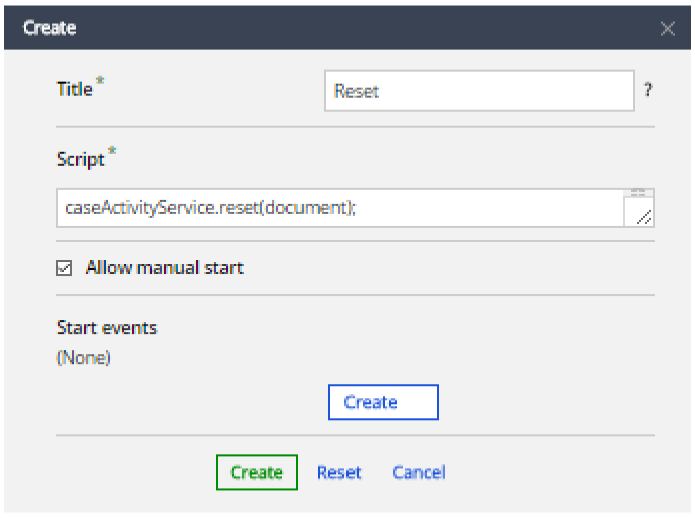
Name: CaseTest
Title: (None)
Description: (None)

Modified by Administrator on Thu 25
Comment

Name: CaseTest
Title: NewTitle
Description: (None)

Figure 18 - The document property has changed

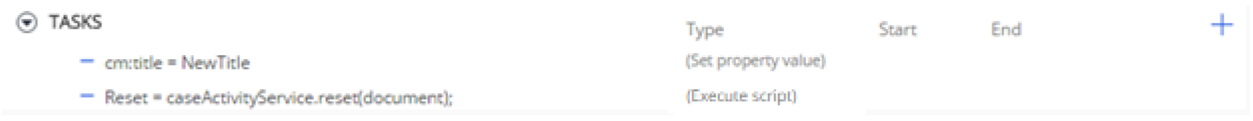
After the operation is completed, the activity becomes uneditable and unrestartable. We will create a new action to reset the completed action to the initial status. For this, we create a new «Start javascript» action and in the javascript-code field in the creation form we enter **caseActivityService.reset(document);** (Figure 19). This action resets all activities in the document to the initial status.


Create

Title *	Reset
Script *	caseActivityService.reset(document);
☑ Allow manual start	
Start events	(None)

Figure 19 - Creation of action for activity reset

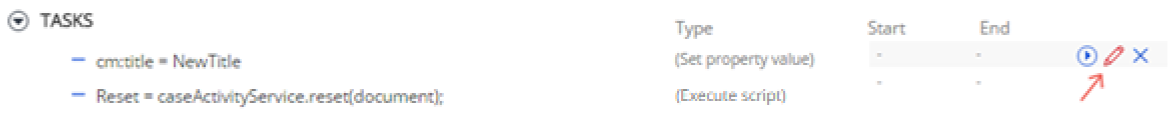
We start the javascript action and we can see that the start and end dates of our first activity have disappeared (Figure 20).


TASKS

	Type	Start	End
cm:title = NewTitle	(Set property value)		
Reset = caseActivityService.reset(document);	(Execute script)		

Figure 20 - The activities status has reset

Now we can edit the "set document property" action. For this, we click the button marked in Figure 21.


TASKS

	Type	Start	End
cm:title = NewTitle	(Set property value)	-	-
Reset = caseActivityService.reset(document);	(Execute script)	-	-

Figure 21 - "Change" button

In the "Value" field we enter a new arbitrary value, click the "Save" button and start the action. As a result, we can see that the "Title" field has taken a new value.

With the "Reset" action demonstrated above one can very conveniently adjust existing and create new processes in case management.

3.2 Stage

Stage is an activity mainly characterized by the capability to incorporate other activities (including other stages as well). In a stage, we can indicate the "Document Status" parameter, which will be set for the current document (case) at stage start.

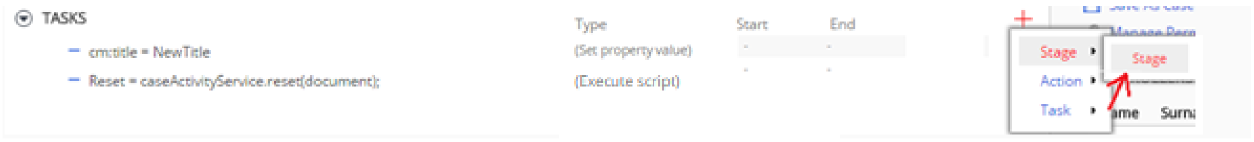
Properties: Title, Description, Document status, Planned start date, Planned end date.

Reaction to events: stage start (start event), stage restart (restart event), stage stop (end event).

Business requirements: If a document (case) contains business requirements, we can select the needed requirements for start or end. If the specified requirements are not observed, the stage cannot change its status.

Example:

We select stage from the new activity menu (Figure 22).

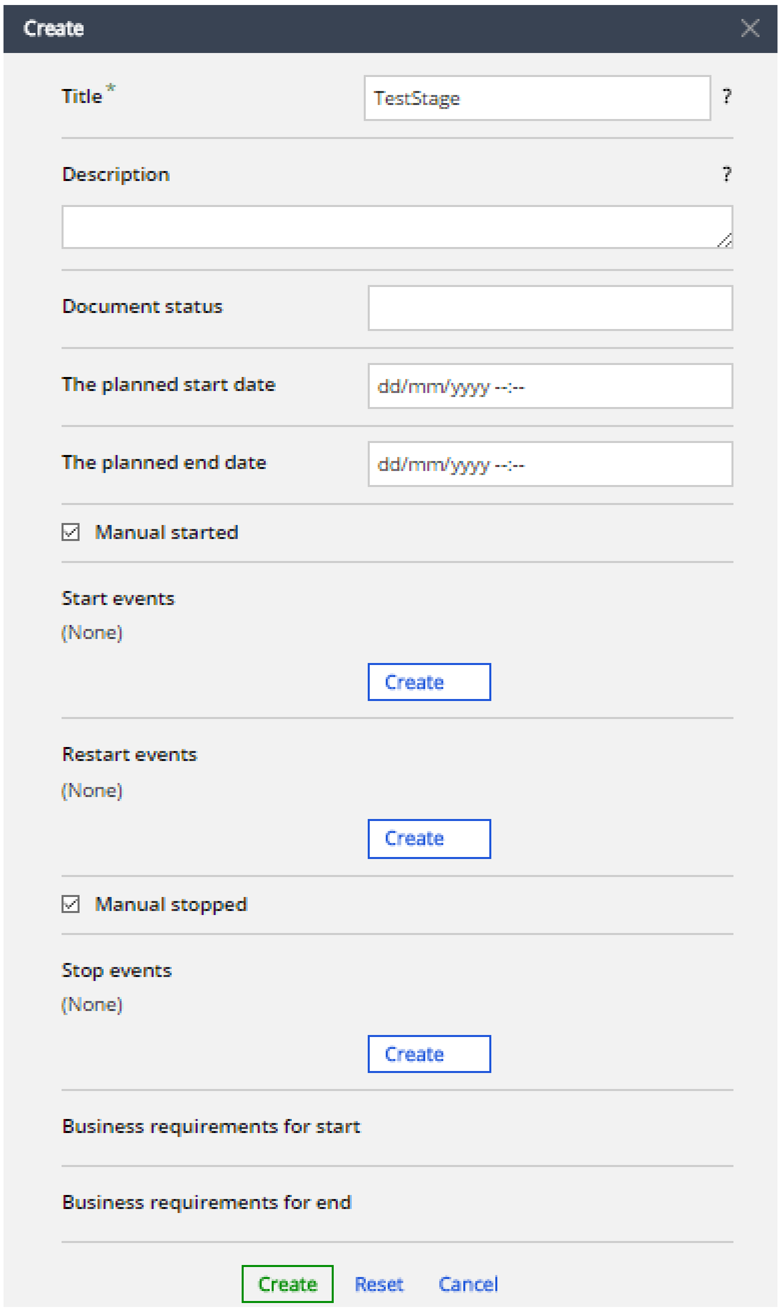


TASKS

- cm:title = NewTitle — (Set property value)
- Reset = caseActivityService.reset(document); — (Execute script)

Figure 22 - Stage creation

In the appearing form, we fill in an arbitrary title and click the green "Create" button at the bottom of the form (Figure 23).

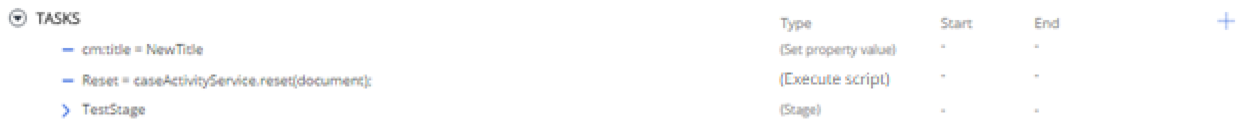


Create

Title *	TestStage
Description	
Document status	
The planned start date	dd/mm/yyyy --:--
The planned end date	dd/mm/yyyy --:--
☑ Manual started	
Start events	(None)
Restart events	(None)
☑ Manual stopped	
Stop events	(None)
Business requirements for start	
Business requirements for end	

Figure 23 - New stage creation

After the stage is created, it will appear in the "Tasks" cardlet (Figure 24).

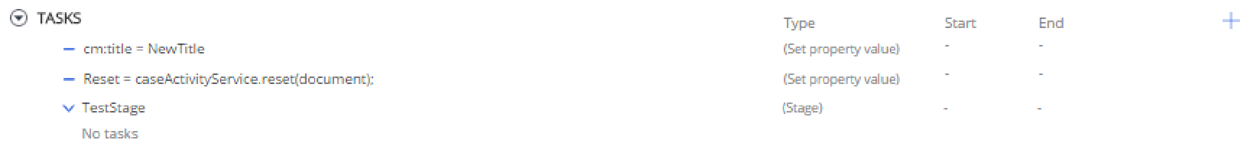


TASKS

	Type	Start	End
cm:title = NewTitle	(Set property value)	-	-
Reset = caseActivityService.reset(document);	(Execute script)	-	-
TestStage	(Stage)	-	-

Figure 24 - New "TestStage"

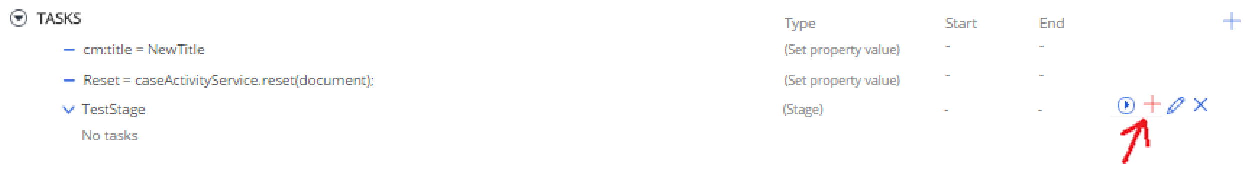
The blue right arrow means the stage is currently "collapsed" and its contents are not shown. If we left-click on the stage, the arrow will point down and the stage will "expand" (Figure 25).



TASKS	Type	Start	End
cm:title = NewTitle	(Set property value)	-	-
Reset = caseActivityService.reset(document);	(Set property value)	-	-
TestStage	(Stage)	-	-
No tasks			

Figure 25 - "Expanded" stage

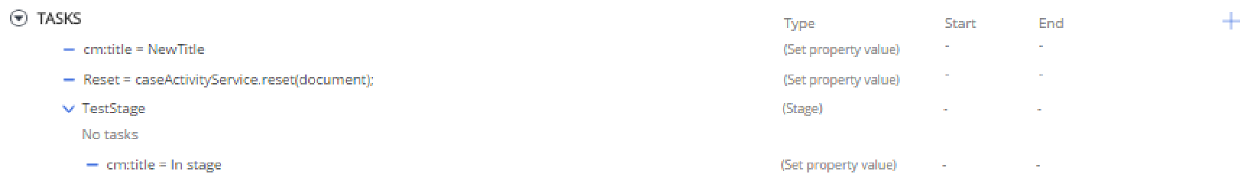
As we can see, currently the stage contains no tasks. We will try to create them. For this, we need to find the "+" button opposite the stage name (Figure 26). After clicking on it we will see a list similar to the already known "+" button in the upper right corner of the "Tasks" cardlet. The difference will be in that new activities will be created inside the stage.



TASKS	Type	Start	End
cm:title = NewTitle	(Set property value)	-	-
Reset = caseActivityService.reset(document);	(Set property value)	-	-
TestStage	(Stage)	-	-
No tasks			

Figure 26 - "Create In-Stage Activity" button

We create a new activity following the already familiar example in Section 3.1 "Set Document Property" and obtain the following result:



TASKS	Type	Start	End
cm:title = NewTitle	(Set property value)	-	-
Reset = caseActivityService.reset(document);	(Set property value)	-	-
TestStage	(Stage)	-	-
No tasks			
cm:title = In stage	(Set property value)	-	-

Figure 27 - In-stage action

By default, the link between a stage and internal activities consists only in nesting (capability to show/hide the stage contents). When using events and conditions, many other opportunities are offered which we will explain further below.

3.3 Task

Task is an activity requiring human interference. It is based on simple BPMN-processes which start at task start and upon their completion the task ends automatically. Task types differ by the BPMN- processes behind them and a required process is selected based on the purpose. If a simple task is needed, containing a comments filed and two buttons - "Confirm" and "Reject", then "Parallel Approval" can be used.

Q. Why only parallel? What if consecutive tasks need to be assigned?

A. Parallel approval is a single task allowing for selection of several approvers within it. If a consecutive process is required, then several approval tasks can be created with a single approver in each. By manipulating start events any types of tasks can be used both for consecutive and for parallel processes.

Reaction to events: task start (start event), task restart (restart event).

Notes: If an incomplete task is restarted, then the process that was started is cancelled and a new process is started.

Business requirements: If a document (case) contains business requirements, we can select the needed requirements for start or end. If the specified requirements are not fulfilled, the task cannot change its status.

Task types:

- Parallel approval

  Simple approval flow that can be used in many cases where the user is only required to leave a comment on the task and provide a result : "Confirmed" or "Reject". The button names can be changed, but it requires necessary records in*.properties files.

Properties: Title, Planned start date, Planned end date, Priority, Approvers, Initiator review not required, Confirmation with comments allowed, Terminate on first rejection, Approvers rights, Localization prefix for approval options.

Note: The capability to change button names needs to be verified.

- Parallel performing

  Properties: Title, Planned start date, Planned end date, Priority, Performer, Performer rights, Verify results.

- Simple task

  Simple, single-performer tasks:

  - Printing process;
  - Send to scan;
  - Send to sign;
  - Reworking process;
  - Registration;
  - Regulatory compliance assessment;
  - Prolong contract;
  - Send for confirmation;
  - Process payment;
  - Archive;
  - Send for counterparty approval;

  Properties: Title, Planned start date, Planned end date, Priority, Process name, Performer.

Example:

We will add to our case a "Parallel Approval" task within TestStage (Figure 28).

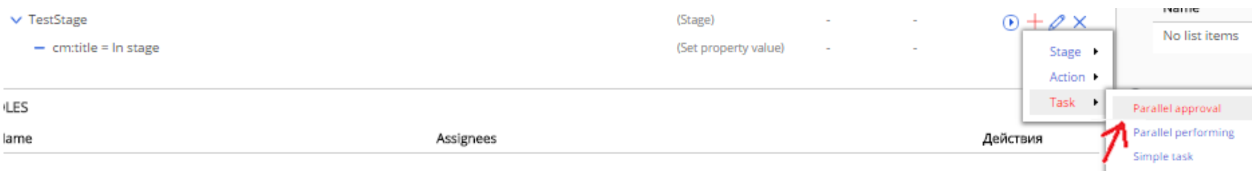
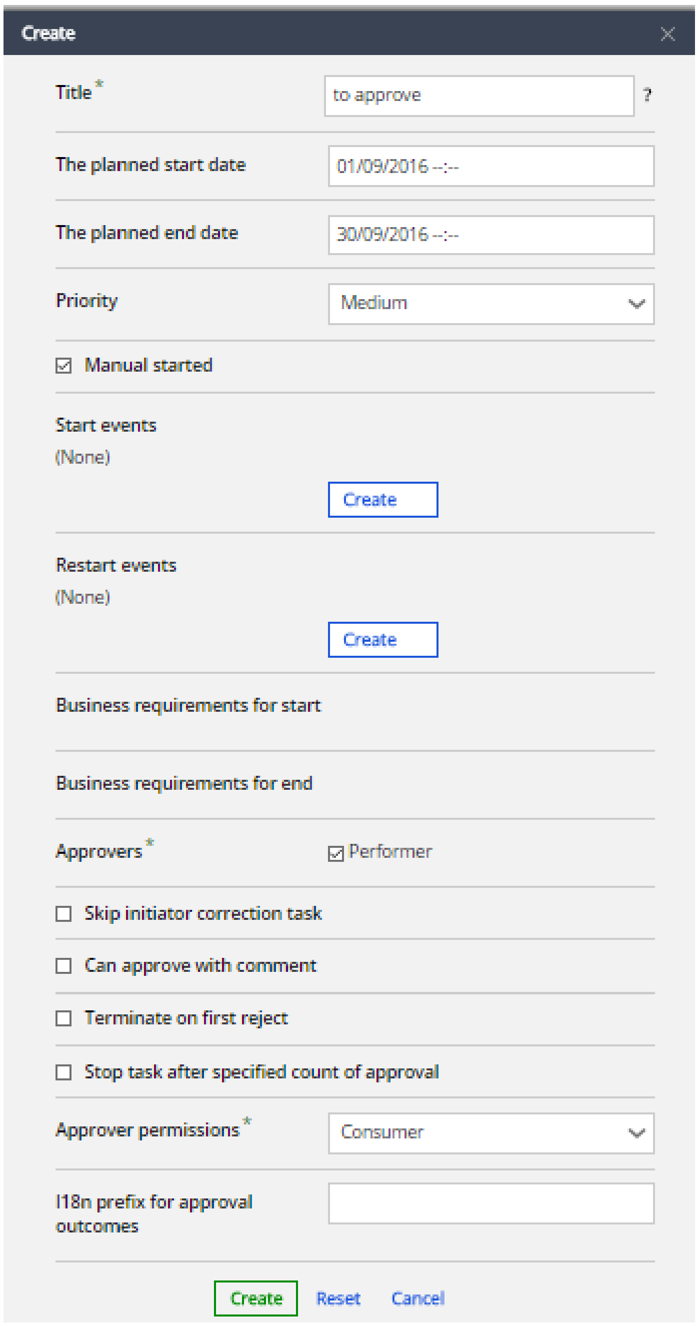


Figure 28 - Creation of the "Parallel Approval" task

We fill in the form as shown in Figure 29.



Create

Title *	to approve
The planned start date	01/09/2016 --:--
The planned end date	30/09/2016 --:--
Priority	Medium

☑ Manual started

Start events
(None)

Restart events
(None)

Business requirements for start

Business requirements for end

Approvers *	☑ Performer

☐ Skip initiator correction task

☐ Can approve with comment

☐ Terminate on first reject

☐ Stop task after specified count of approval

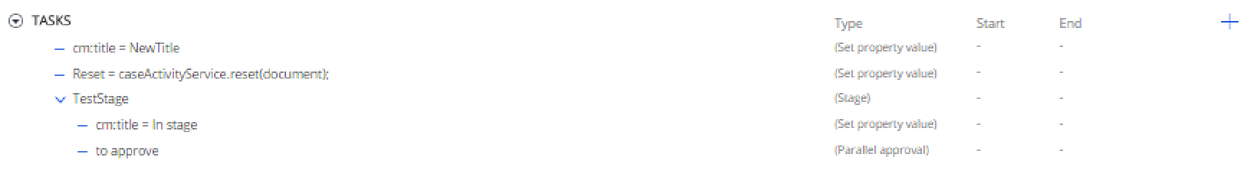
Approver permissions *	Consumer
I18n prefix for approval outcomes	

Figure 29 - Creation form for the "Parallel Approval" task

Note: If no one is selectable on your approvers list, then your case contains no roles. To create a task, at least one role is required.

Note: Make sure you are among the approvers in your selected role (or that you are part of the group specified among the approvers). This will allow you to perform created tasks right in the document (case) card.

We click the green "Create" button at the bottom of the form and we can see that our task has appeared in the "Tasks" cardlet (Figure 30).

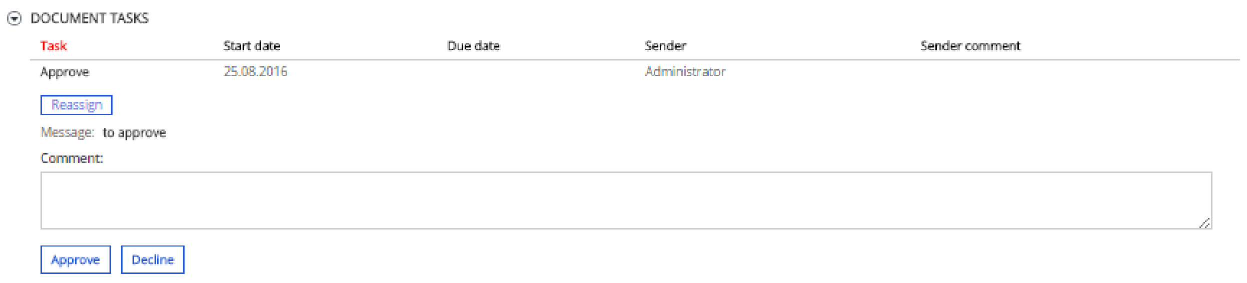


TASKS

	Type	Start	End
– cm:title = NewTitle	(Set property value)	-	-
– Reset = caseActivityService.reset(document);	(Set property value)	-	-
⌄ TestStage	(Stage)	-	-
– cm:title = In stage	(Set property value)	-	-
– to approve	(Parallel approval)	-	-

Figure 30 - New task

We start the task just as we did for actions by clicking the "Start" button opposite the task. We confirm start and, if you are among the approvers, (or in the group specified in the role), then a completable task will appear in the case card (Figure 31).



DOCUMENT TASKS

Task	Start date	Due date	Sender	Sender comment
Approve	25.08.2016		Administrator	

Reassign

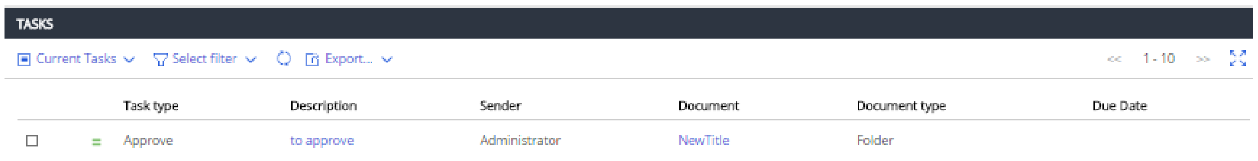
Message: to approve

Comment:

Approve Decline

Figure 31 - "Management Approval" task

Information about the started task will also appear on the dashboard task list or on the "My Tasks" page (Figures 32 and 33).

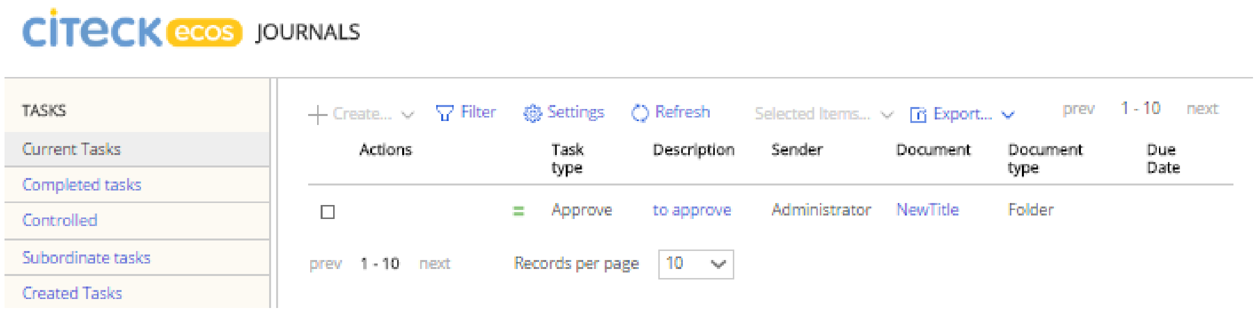


TASKS

Current Tasks ⌄ Select filter ⌄ Export... ⌄ << 1 - 10 >>

		Task type	Description	Sender	Document	Document type	Due Date
☐	=	Approve	to approve	Administrator	NewTitle	Folder	

Figure 32 - "Tasks" dashlet on the main page



citeck ecos JOURNALS

TASKS
Current Tasks
Completed tasks
Controlled
Subordinate tasks
Created Tasks

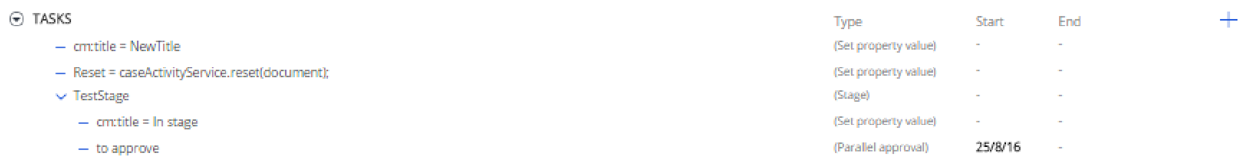
Create... Filter Settings Refresh Selected Items... Export... prev 1 - 10 next

	Actions	Task type	Description	Sender	Document	Document type	Due Date
☐	=	Approve	to approve	Administrator	NewTitle	Folder	

prev 1 - 10 next Records per page 10

Figure 33 - "Current tasks" log

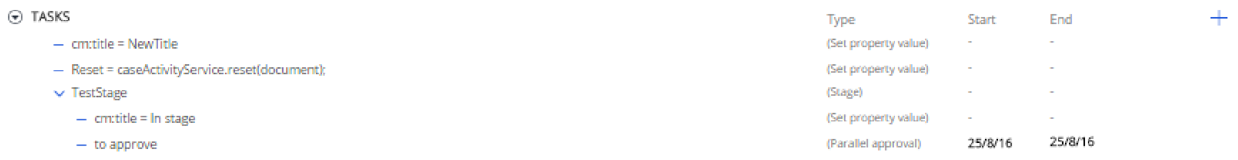
After task start, we may note it now has a start date set which indicates the task has started (Figure 34). The end date is missing since the task is not yet completed.



TASKS	Type	Start	End
– cm:title = NewTitle	(Set property value)	-	-
– Reset = caseActivityService.reset(document);	(Set property value)	-	-
TestStage	(Stage)	-	-
– cm:title = In stage	(Set property value)	-	-
– to approve	(Parallel approval)	25/8/16	-

Figure 34 - "Tasks" cardlet status after the "Management Approval" task is started

Now we will try to complete the process by clicking the "Confirmed" button in the form shown in Figure 31. After this action the "Management Approval" task is automatically ended (Figure 35). Its completion is rigidly linked to the completion of the started BPMN process, therefore, tasks do not allow for being completed based on an event or manually.



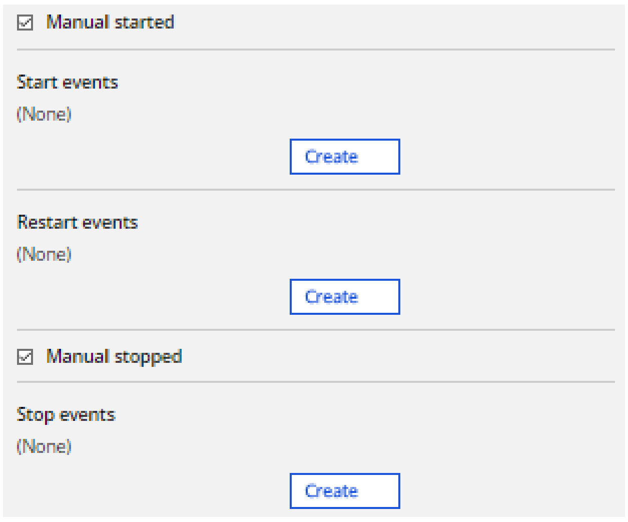
TASKS	Type	Start	End
– cm:title = NewTitle	(Set property value)	-	-
– Reset = caseActivityService.reset(document);	(Set property value)	-	-
TestStage	(Stage)	-	-
– cm:title = In stage	(Set property value)	-	-
– to approve	(Parallel approval)	25/8/16	25/8/16

Figure 35 - Completed "Management Approval" task

4. Events

Events are a flexible tool to interlink stages, tasks and actions into an integral process.

Activities can be started, stopped or restarted in two ways: manually or based on a specified event. When creating or editing activities in the form, we can specify that an activity is to be started manually (Manual started) or stopped manually (Manual stopped) (Figure 36) If we specify that one of these actions is not manual, then we must specify in the form the event based on which the action should be executed.


☑ Manual started

Start events
(None)

Restart events
(None)

☑ Manual stopped

Stop events
(None)

Figure 36 - Activity start, end and restart controls

Technically, events have the following structure (activities used as example):

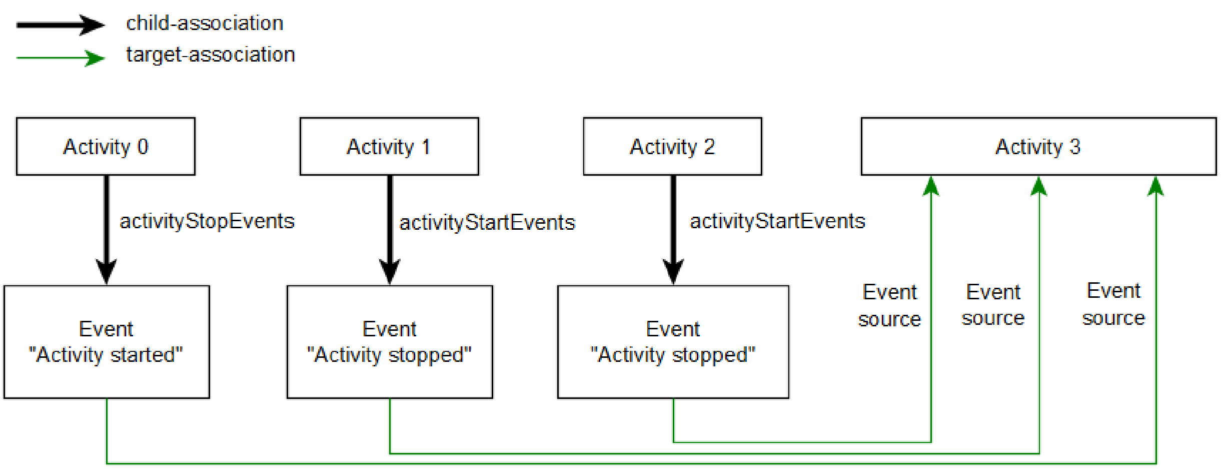


Figure 37 - Interlinks between activities through events

The lifecycle of activity interlinks through events is as follows:

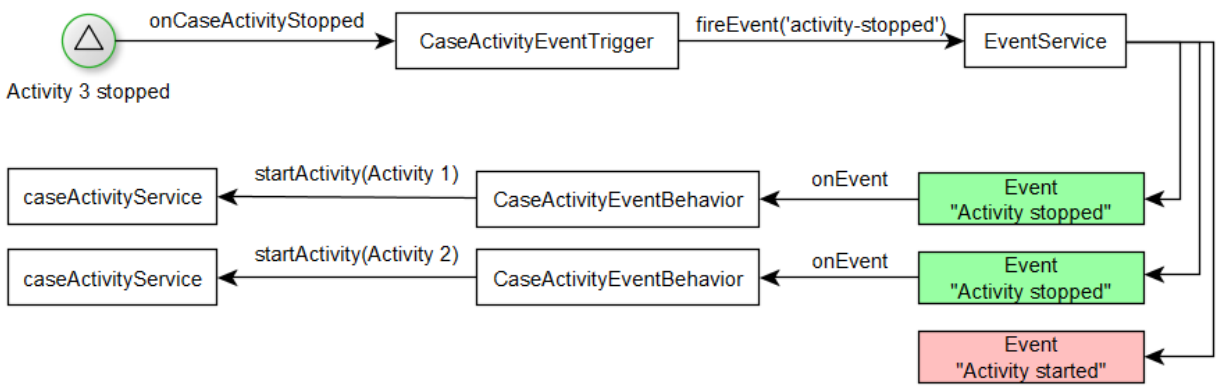


Figure 38 - Lifecycle of activity interlinks through events

Currently, there are three types of reaction to events (Indicated in brackets are field names in edit and creation forms):

- Start activity (start event)
- Restart activity (restart event)
- Stop activity (end event)

When creating or editing an activity, we can specify what actions will lead to its start, restart or end.

Event types:

- **Last active sub-task ended**

  Event occurring when a specified stage has its last active sub-task completed. This event is convenient to use for completing a stage with no active sub-tasks left.

  Event source: neighbour or current stage.

- **Task started**

  Event occurring when a specified activity has started.

  Event source: neighbour activities, current activity, parent activity.

- **Task ended**

  Event occurring when a specified activity has been completed.

  Event source: neighbour activities, current activity, parent activity.

- **User actions**

Event occurring when a user has clicked on a corresponding action on the action list in the document (case) card. Adding such elements to any activity in a case will lead to a new action appearing in the "Actions" column.

Event type: <u>identifier used to determine equality or inequality of different actions of a user. One event type corresponds to one action in the "Actions" column.</u>

Roles: <u>roles for which a specified action is available.</u>

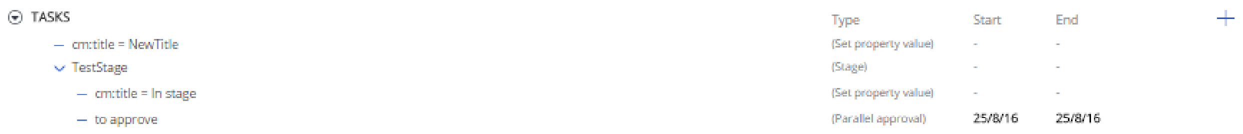
Event source: <u>current document (case)</u>

- **Case created**

Event occurring when a new case has been created. This event can be used, e.g. to start a process after case creation or to perform some initialization actions (e.g., initial status setting).

Example:

We will interlink our activities within the created case through the event mechanism. We will use the activities we created in the previous examples as a starting point. These are presented in Figure 39.



TASKS	Type	Start	End
— cm:title = NewTitle	(Set property value)	-	-
TestStage	(Stage)	-	-
— cm:title = In stage	(Set property value)	-	-
— to approve	(Parallel approval)	25/8/16	25/8/16

Figure 39 - Case activities

We edit "TestStage" by clicking the button marked in Figure 40.

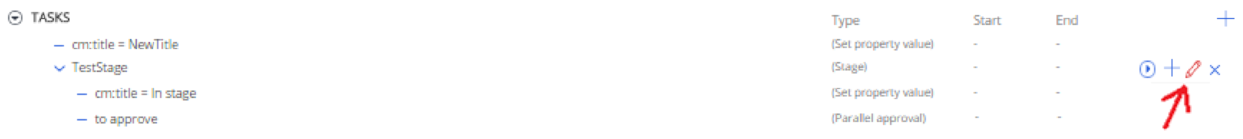


Figure 40 - "Edit Activity" button

We uncheck the Manual Started checkbox (not necessarily), click the Create button in the "Start Event" field and select "Task Ended" (Figure 41).

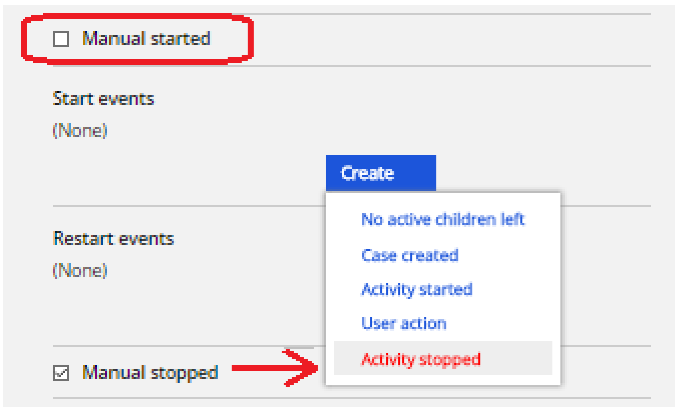


Figure 41 - Creation of an event for stage start

We select our first action as the event source (Figure 42) and leave the "Conditions" field empty.

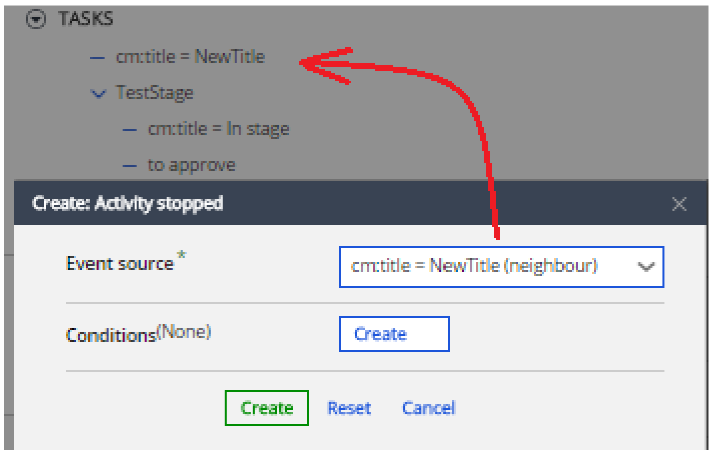


Figure 42 - Creation of the "Task Ended" event

Note: If an activity from the case is missing on the drop-down list of event sources, we will need to refresh the webpage and try again. If the activity is still missing, then this may be indicative of its unavailability in the current context. Example: for the "Task Ended" event, only neighbour activities and the parent can be used as the source.

Note: Please note that the selected activity title is labelled as "neighbour". This is additional information about the relation of the selected activity to the current one. Possible options:

- *neighbour – neighbour activity;*
- *self – current activity;*
- *parent – parent activity.*

We click the Create button and create an end event for our stage. We select "Last active sub-task ended" as the event type (Figure 43).

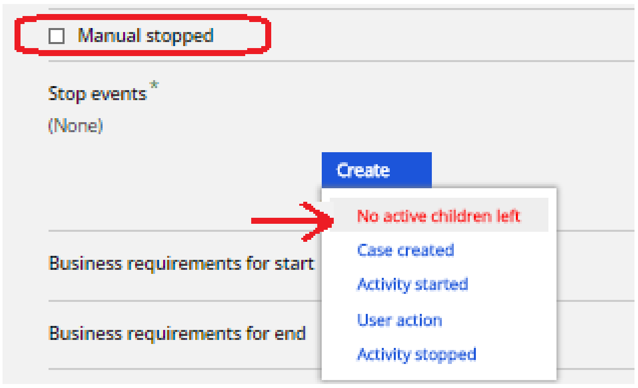
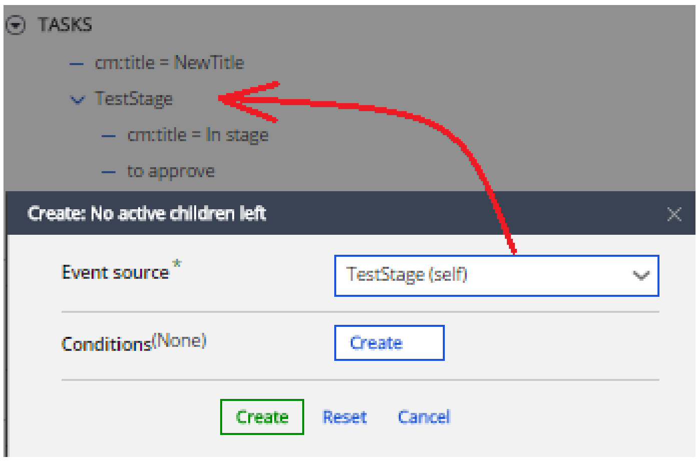


Figure 43 - End event for a stage

We select the current stage as the event source (Figure 44).



TASKS

— cm:title = NewTitle

⌄ TestStage

— cm:title = In stage

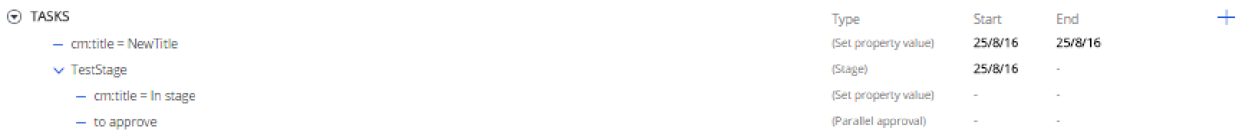
— to approve

Create: No active children left

Event source *	TestStage (self)
Conditions(None)	Create

Figure 44 - Creation of an end event

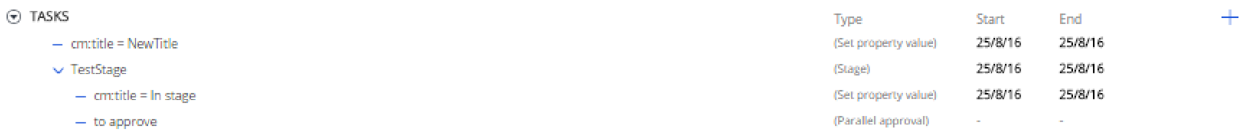
We click on "Create" in the event creation form and save the changes in the stage. Now we will try to start our first activity - action (cm:title = NewTitle) - and we can see the following result:



TASKS	Type	Start	End
— cm:title = NewTitle	(Set property value)	25/8/16	25/8/16
⌄ TestStage	(Stage)	25/8/16	-
— cm:title = In stage	(Set property value)	-	-
— to approve	(Parallel approval)	-	-

Figure 45 - Result of action start outside the stage

As we can see, the stage was started automatically after completion of the started action. Now we will try to trigger stage end. For this, we start our action in-stage («cm:title = In stage»). The result is presented in Figure 45.

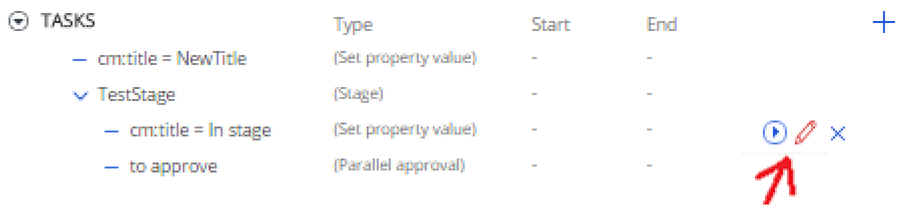


TASKS	Type	Start	End
— cm:title = NewTitle	(Set property value)	25/8/16	25/8/16
⌄ TestStage	(Stage)	25/8/16	25/8/16
— cm:title = In stage	(Set property value)	25/8/16	25/8/16
— to approve	(Parallel approval)	-	-

Figure 45 - Result of action start in-stage

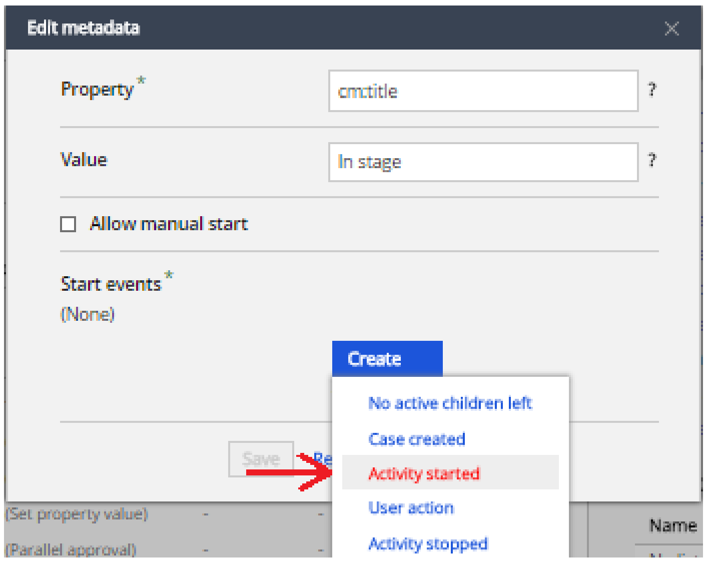
When the in-stage action is completed, the system has checked for other started in-stage activities and upon finding none, the stage was ended.

We reset the case status and, as a next step, add a start event to in-stage activities following the already known scheme (Figures 46, 47 and 48). Event type: "Task started", event source: TestStage.



TASKS	Type	Start	End
— cm:title = NewTitle	(Set property value)	-	-
TestStage	(Stage)	-	-
— cm:title = In stage	(Set property value)	-	-
— to approve	(Parallel approval)	-	-

Figure 46 - In-stage action editing



Edit metadata

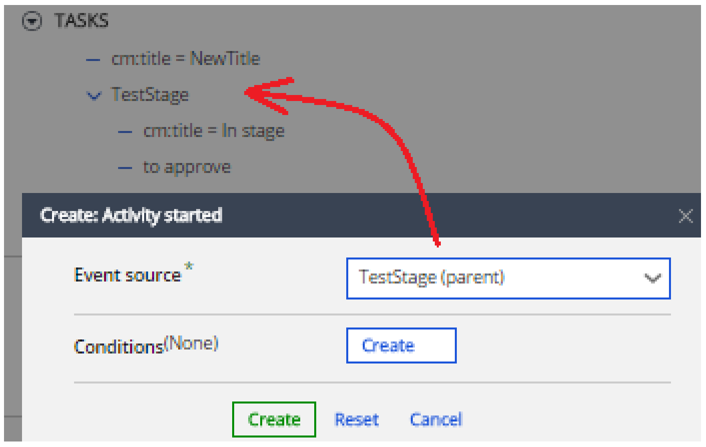
Property * : cm:title

Value: In stage

☐ Allow manual start

Start events *
(None)

Figure 47 - Creation of the "Task Started" event



TASKS
— cm:title = NewTitle
TestStage
— cm:title = In stage
— to approve

Create: Activity started

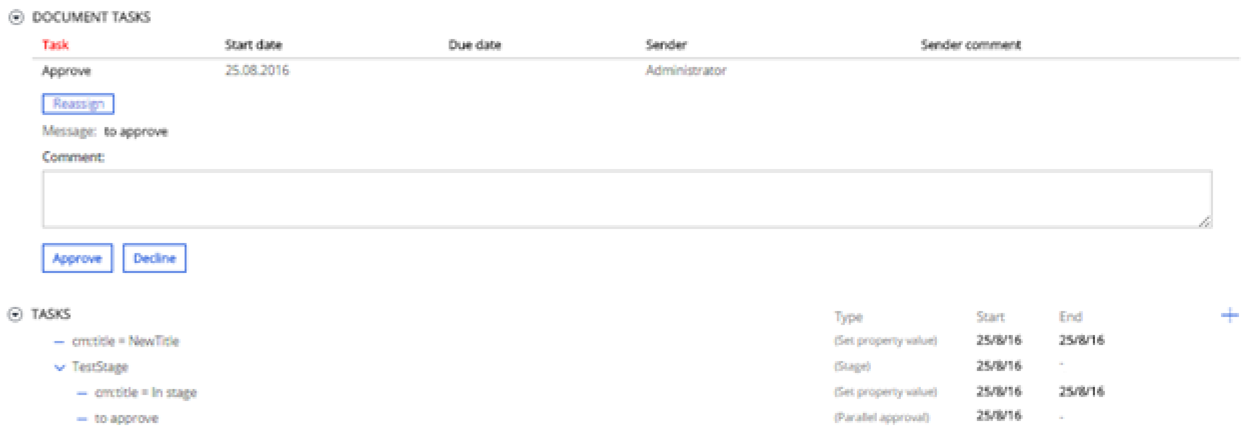
Event source * : TestStage (parent)

Conditions (None)

Figure 48 - We select the parent stage as the event source

Now we perform the same actions for the "Management Approval" task.

Now we will test the created process. For this, we start the first action outside the stage (cm:title = NewTitle) and see the following result:



DOCUMENT TASKS

Task	Start date	Due date	Sender	Sender comment
Approve	25.08.2016		Administrator	

Message: to approve

Comment:

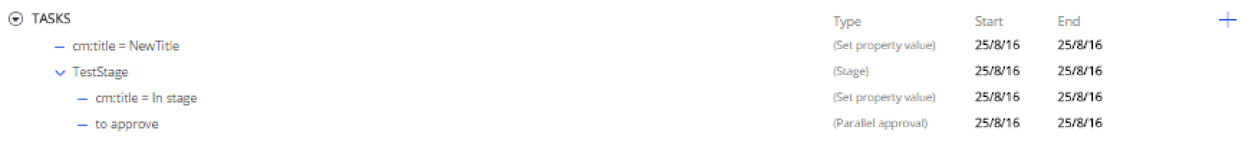
TASKS

	Type	Start	End
cm:title = NewTitle	(Set property value)	25/8/16	25/8/16
TestStage	(Stage)	25/8/16	-
cm:title = In stage	(Set property value)	25/8/16	25/8/16
to approve	(Parallel approval)	25/8/16	-

Figure 49 - Result of action start outside the stage

Note: if upon start you did not have the "Document Tasks" cardlet appear, then you have not been assigned the role selected during creation of the "Management Approval" task.

We click on "Confirmed" in the form of the started process, and the case activities now look as shown in Figure 50.

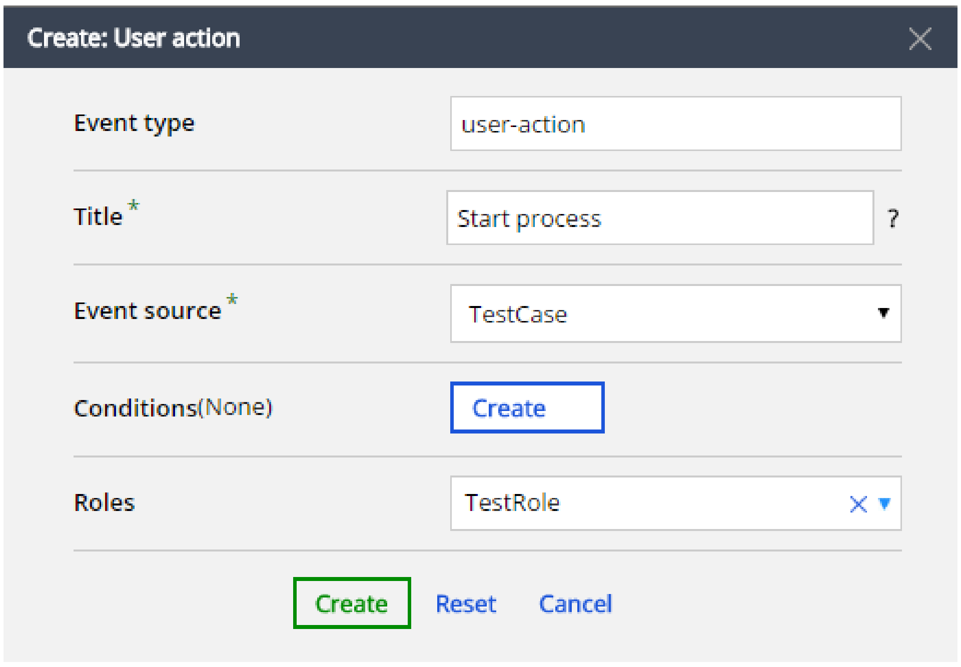


TASKS	Type	Start	End
– cm:title = NewTitle	(Set property value)	25/8/16	25/8/16
⌄ TestStage	(Stage)	25/8/16	25/8/16
– cm:title = In stage	(Set property value)	25/8/16	25/8/16
– to approve	(Parallel approval)	25/8/16	25/8/16

Figure 50 - Ended process

As we can see, all activities we interlinked with each other have completed their operation and the process can be regarded as completed.

We will add the option to start the process through the actions column. For this, we will need a "User Action" event. We go to the «cm:title = NewTitle» action editor and add a start event with the "User Action" type. We select the title and source of the event (only case is available) and the role for which this action will be available (Figure 51).



Create: User action

Event type	user-action
Title *	Start process
Event source *	TestCase
Conditions(None)	Create
Roles	TestRole

Figure 51 - Creation of a "User Action" event

After creation of the new action, we refresh the page and we can see a new record in the "Actions" column (Figure 52).

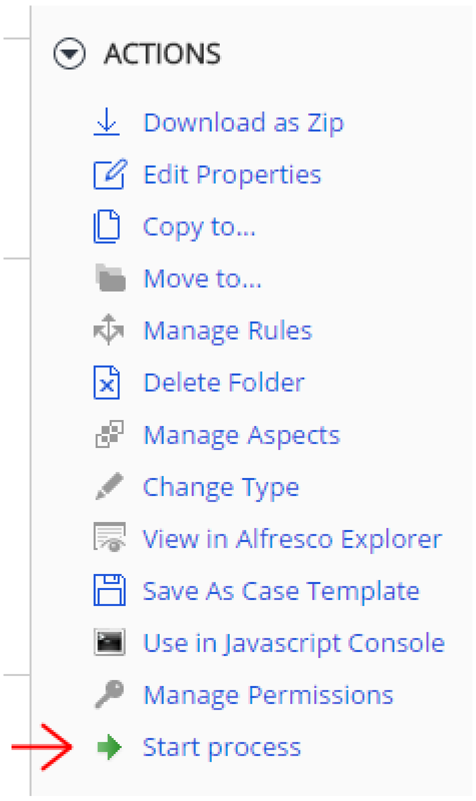


Figure 52 - New action

We click on Start process and we can see that our process has started, but the action is still selectable and if we try to restart it without resetting the case status, this may result in absence of any actions or in error message. To avoid this, we can add a javascript condition for user action, with the following content:

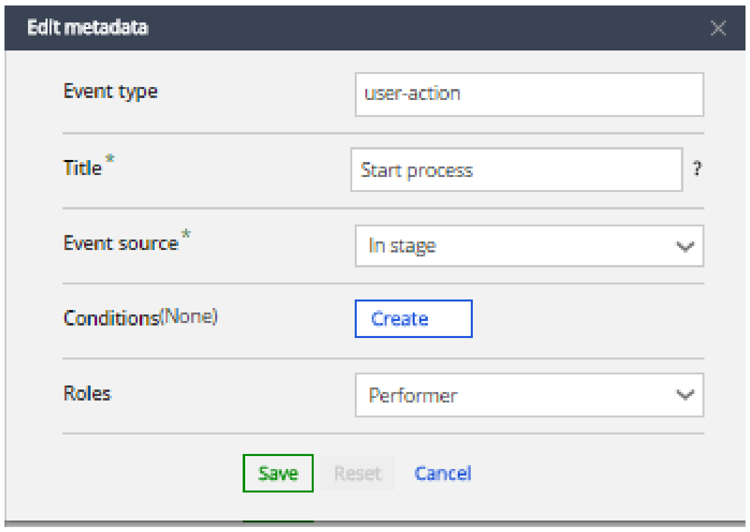
```
event.parent.properties['lc:state'] == 'Not started'
```

Thus, after the activity linked to the event changes its status, the user action will disappear from the "Actions" column.

Event conditions are dwelled on in detail in the next section below.

5. Event Conditions

Event conditions are a tool enabling process branching. When creating or editing any event, we can specify conditions to be fulfilled for the event to occur (Figure 53). If the conditions are multiple, they are integrated through the logic gate AND.


Edit metadata

Event type	user-action
Title *	Start process
Event source *	In stage
Conditions(None)	Create
Roles	Performer

Figure 53 - Event edit form

Condition types:

- Compare property

Compares a specified property of a document (case) with a specified value. Various conditions for property and value correspondence can be specified (operation): equals, contains, begins with, ends with, greater than, greater than or equals, less than, less than or equals.

Properties: Property, Operation, Value

- Evaluate javascript

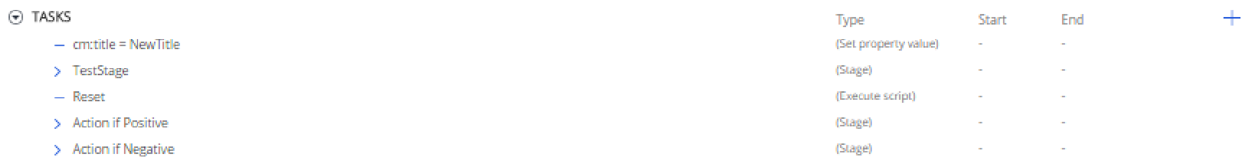
Evaluation of an arbitrary javascript code which should return either true or false. In the context, a **process** object is available, which contains all the variables of a newly completed process. Thus, based on their values, an event occurrence decision can be made.

The context also contains an **event** object by which the event source or the activity reacting to the event can be retrieved (this functionality has not yet been tested).

Properties: javascript expression

Example:

As an example of condition usage we will create two stages neighbouring on TestStage and name them "Action if Positive" and "Action if Negative" (Figure 54).



TASKS	Type	Start	End
cm:title = NewTitle	(Set property value)	-	-
TestStage	(Stage)	-	-
Reset	(Execute script)	-	-
Action if Positive	(Stage)	-	-
Action if Negative	(Stage)	-	-

Figure 54 - Appearance of the "Tasks" cardlet after stage addition

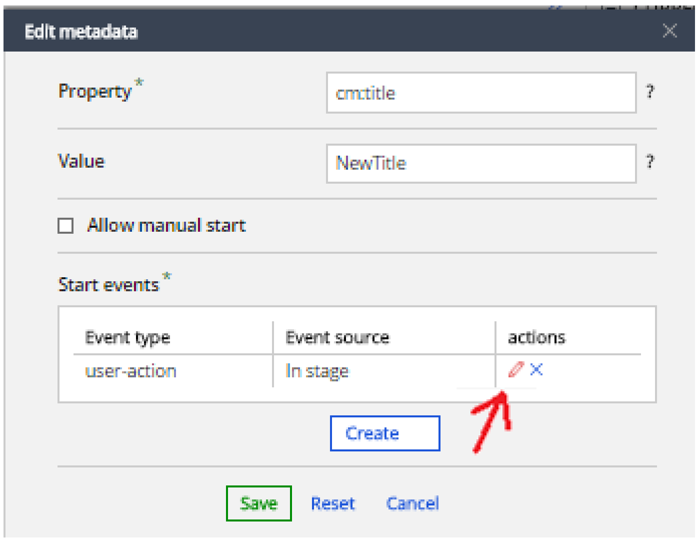
For the new stages, we will add "Start Event" with the "Task Ended" type and a TestStage event source in the same manner as we did before.

We will use the approval result in the "Management Approval" task as the branching condition.

The results of parallel approval can be the following values:

- Confirmed
- ConfirmedWithComment;
- Reject

We will add conditions for start events for new stages based on which one stage will be started after a successful confirmation, and the other one after rejection. For this, we will go to the edit manager for start events (Figure 55) and will create a javascript condition (Figure 56).



Edit metadata

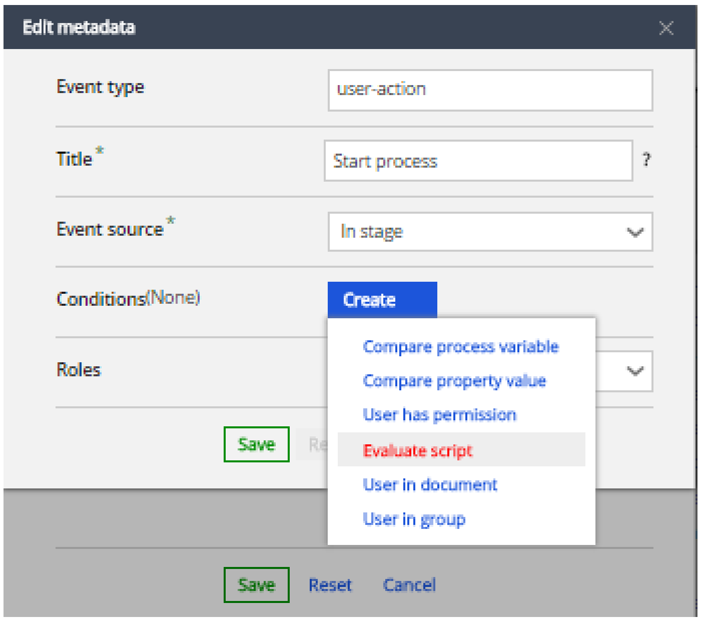
Property *	cm:title
Value	NewTitle

☐ Allow manual start

Start events *

Event type	Event source	actions
user-action	In stage	

Figure 55 - Event editing



Edit metadata

Event type	user-action
Title *	Start process
Event source *	In stage
Conditions(None)	
Roles	

Figure 56 - Creation of a start condition

The result of completion of the "Parallel Approval" task will be written in the **wfcf_confirmOutcome** variable, and all process variables will be written in the **process** object.

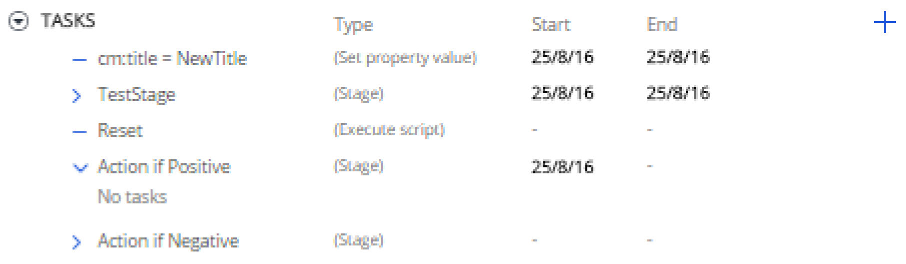
For the "Action if Positive" stage we enter the following javascript expression in the start event condition:

```
process.wfcf_confirmOutcome=='Confirmed' ||
process.wfcf_confirmOutcome=='ConfirmedWithComment'
```

And for the "Action if Negative" stage we will check "rejecting" operation of the process:

```
process.wfcf_confirmOutcome=='Reject'
```

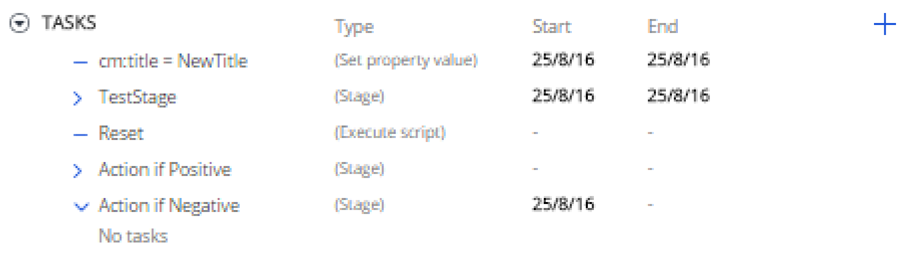
We save the changes and start the first action outside the stage (cm:title = NewTitle). As before, this should lead to start of the "Management Approval" task (also, TestStage can be set to manual start and started manually). We click the "Confirm" button and our process will look like this:



TASKS	Type	Start	End
cm:title = NewTitle	(Set property value)	25/8/16	25/8/16
TestStage	(Stage)	25/8/16	25/8/16
Reset	(Execute script)	-	-
Action if Positive	(Stage)	25/8/16	-
No tasks			
Action if Negative	(Stage)	-	-

Figure 57 - The "Confirm" button is clicked.

We will reset activities status in the case and will try once again, but this time clicking on "Reject". The result is presented in Figure 58.



TASKS	Type	Start	End
— cm:title = NewTitle	(Set property value)	25/8/16	25/8/16
> TestStage	(Stage)	25/8/16	25/8/16
— Reset	(Execute script)	-	-
> Action if Positive	(Stage)	-	-
⌄ Action if Negative	(Stage)	25/8/16	-
No tasks			

Figure 58 - The "Reject" button is clicked.

As we can see, thanks to the conditions only one stage is started depending on the approver's decision.

6. Business Requirements

Business requirements is a mechanism enabling setting of requirements for a case at a specified process stage. For example, upon completion of the "Credit Committee Meeting" task a "Credit Committee MoM" can be requested, and the task cannot be completed until the document is attached to the case.

Business requirements are created globally for the whole system, and in a case, selection is made from among these requirements. Creation and editing of business requirements is done through the business requirements log. To go to the log, we should select "Administrator Tools" in the upper menu and click on the "System Logs" item (Figure 59).

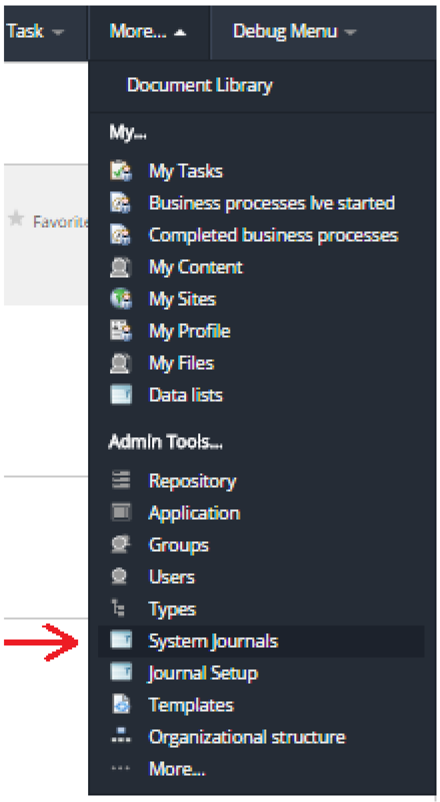


Figure 59 - System Logs

If such a menu item is absent, we can go to the system logs via a direct link: **<address>:<port>/share/page/journals2/list/system**.

Further, we go to the "Case Business Requirements" log:

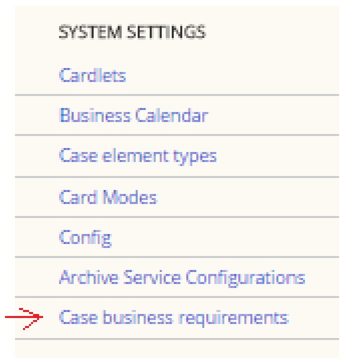


Figure 60 - Selection of the "Case Business Requirements" log

If you have a clean system installed your log will be empty. We will create a new business requirement. For this, we click the "Create" button and select the "Set of Requirements" item:

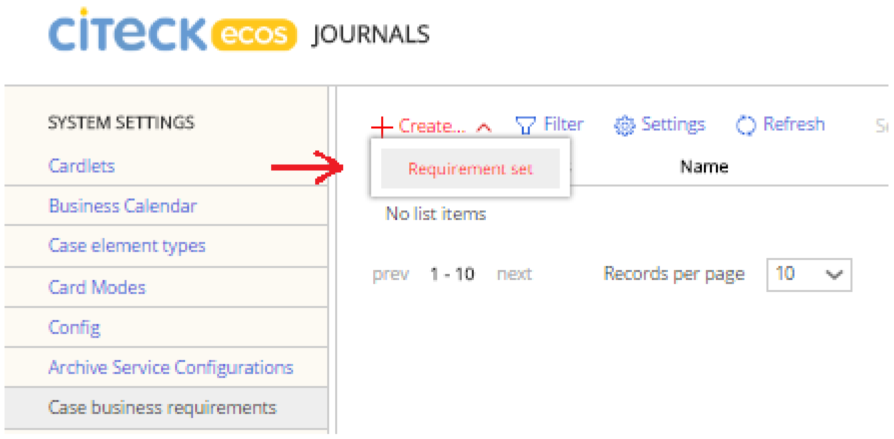
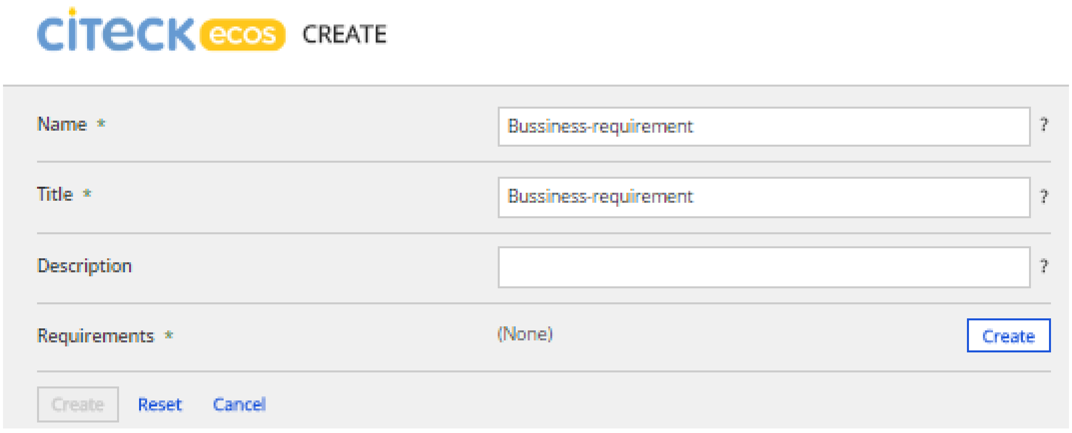


Figure 61 - Creation of a new set of requirements

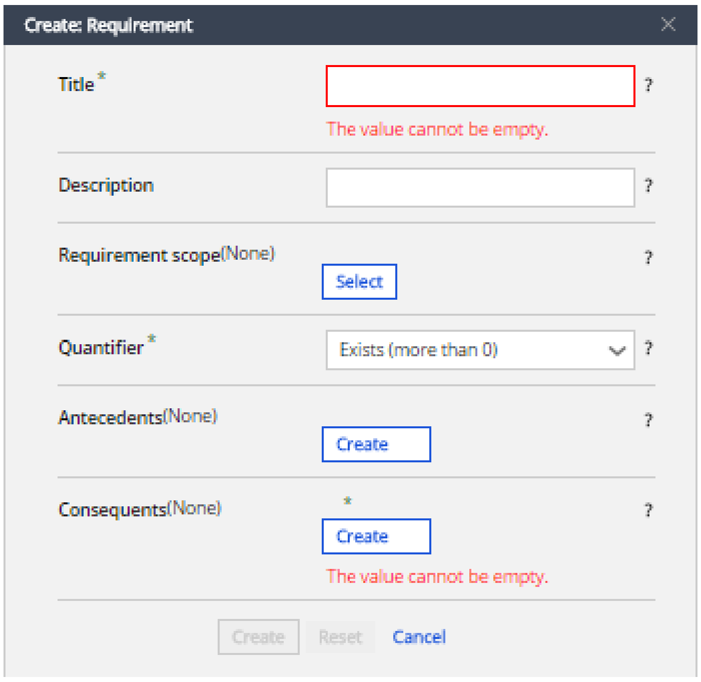
The creation form for a set of requirements is presented in Figure 62.


CREATE

Name *	Bussiness-requirement	?
Title *	Bussiness-requirement	?
Description		?
Requirements *	(None)	Create

Figure 62 - Creation of a set of requirements

We will add new requirements to our set. After clicking the "Create" button, we will see the following form in the "Requirements" field:


Create: Requirement

Title *	The value cannot be empty.	?
Description		?
Requirement scope(None)	Select	?
Quantifier *	Exists (more than 0)	?
Antecedents(None)	Create	?
Consequents(None)	* Create The value cannot be empty.	?

Figure 63 - Creation form for a requirement

The "Title" and "Description" fields can be filled in arbitrarily.

The "Scope" field defines, correspondingly, the validity scope of requirements.

Available scopes:

- Types of case elements
- Categories
- Tags
- Additional files
- Documents
- Events
- Tasks
- Roles
- Linked documents
- Sub-cases

We select the "Documents" scope.

The "Quantity" field defines how a requirement will be verified. E.g., if we have a requirement for a document, we can specify in this field:

- Exists: the document should be enclosed in the case as a single copy or multiple copies (quantity >=1);
- Single: it can exist in the case as a single copy (quantity 0 or 1);
- Equals 0 - the document should not be enclosed in the case;
- Equals 1: the document should be enclosed in the case as a single copy;
- Optional: the requirements are deemed fulfilled in any case.

The "Prerequisites" field is completed with requirements, which define whether the requirements described in the "Requirements" field should be verified. In the "Prerequisites" and "Requirements" fields, identical sets of creatable kinds of requirements are available.

Kinds of requirements:

- Type requirements

- Sub-case type is required
- Condition
- JavaScript predicate
- Set of requirements is fulfilled
- Requirement

The mechanism of requirement verification is roughly as follows:

For any element within the scope: Evaluate the prerequisites. If all of them return true or are simply absent, we go to requirement verification. If at least one prerequisite returned false, the parent requirement is deemed fulfilled and the requirements form the "Requirements" field are not verified. Thus, the quantity of compliant elements is evaluated and this quantity is compared with the "Quantity» field. If the quantity is appropriate, then the parent requirement is deemed fulfilled, otherwise - not.

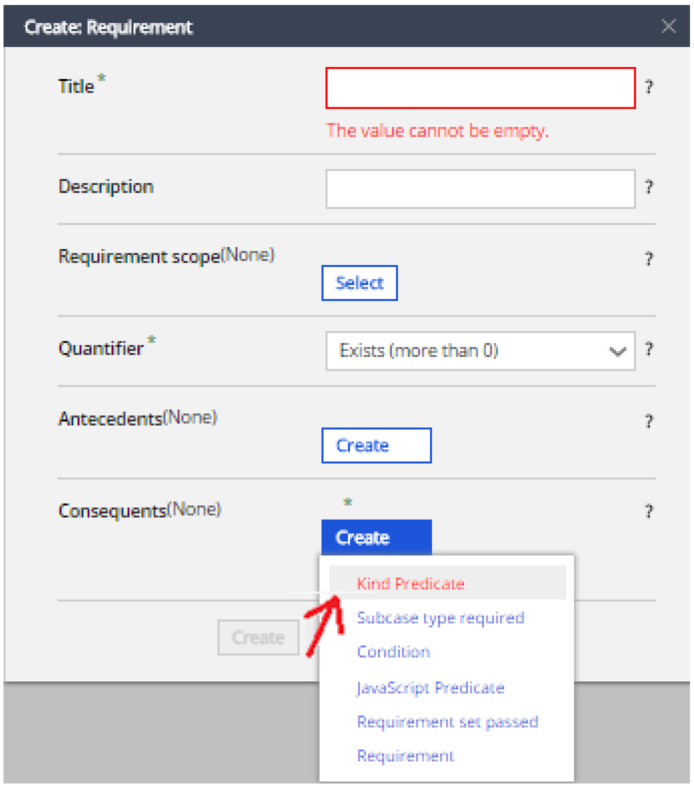
Verification of prerequisites and requirements can be written as follows:

$$p_1 \ \& \ p_2 \ \& \ p_3 \ || \ r_1 \ \& \ r_2 \ \& \ r_3$$

where $p_{1..3}$ – prerequisites;

$r_{1..3}$ – requirements.

We will create a type requirement:


Create: Requirement

Title *

The value cannot be empty.

Description

Requirement scope(None)

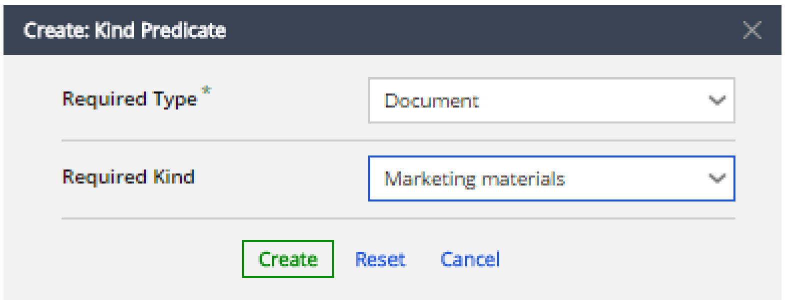
Quantifier * Exists (more than 0)

Antecedents(None)

Consequents(None)

Figure 64 - Creation of a type requirement

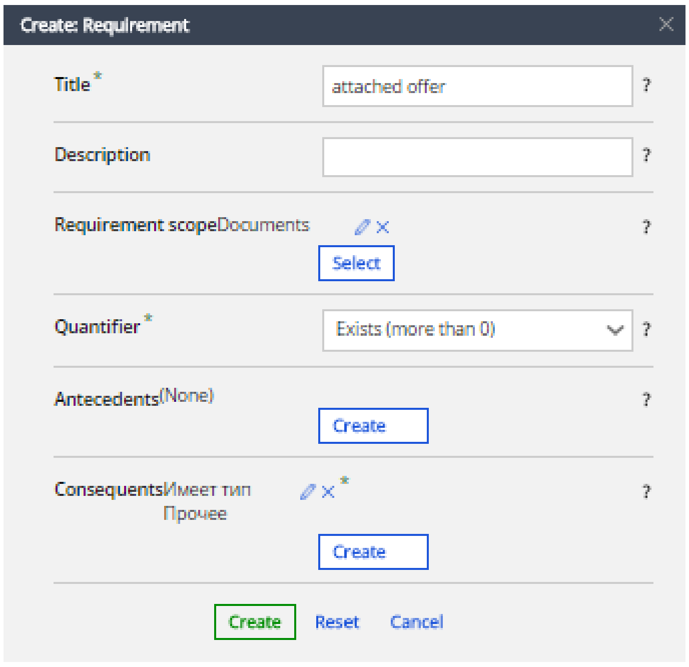
We select "Document" as the required type and " Marketing materials" as the required kind:


Create: Kind Predicate

Required Type *	Document
Required Kind	Marketing materials

Figure 65 - Creation of a type requirement

We click the create button and our creation form should look as shown in Figure 66:


Create: Requirement

Title *	attached offer
Description	
Requirement scope	Documents
Quantifier *	Exists (more than 0)
Antecedents	(None)
Consequents	Имеет тип Прочее

Figure 66 - Requirement creation form after the fields are completed

We click on create and complete creating our set of requirements similarly by clicking the "Create" button

We go to the case card and open the "Case Setting" tab (Figure 67).

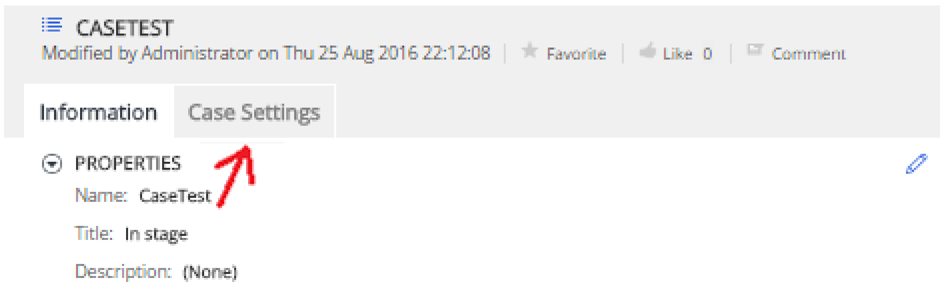


Figure 67 - "Case Setting" tab

Then we click the "Add Business Requirement" button (Figure 68).

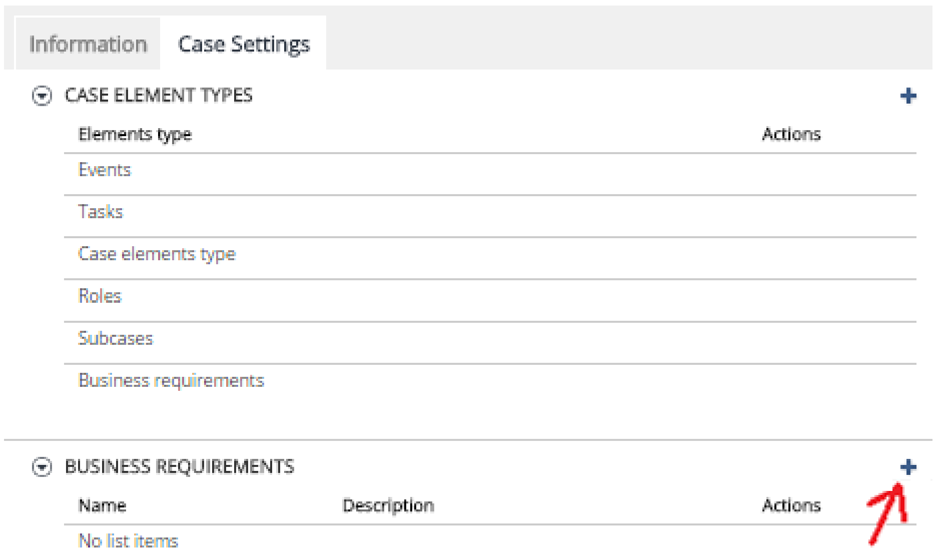


Figure 68 - "Add Business Requirement" button

In the opening window, we can see our business requirement we created earlier (Figure 69). We click the "+" button near the requirement and confirm selection by clicking the "OK" button.

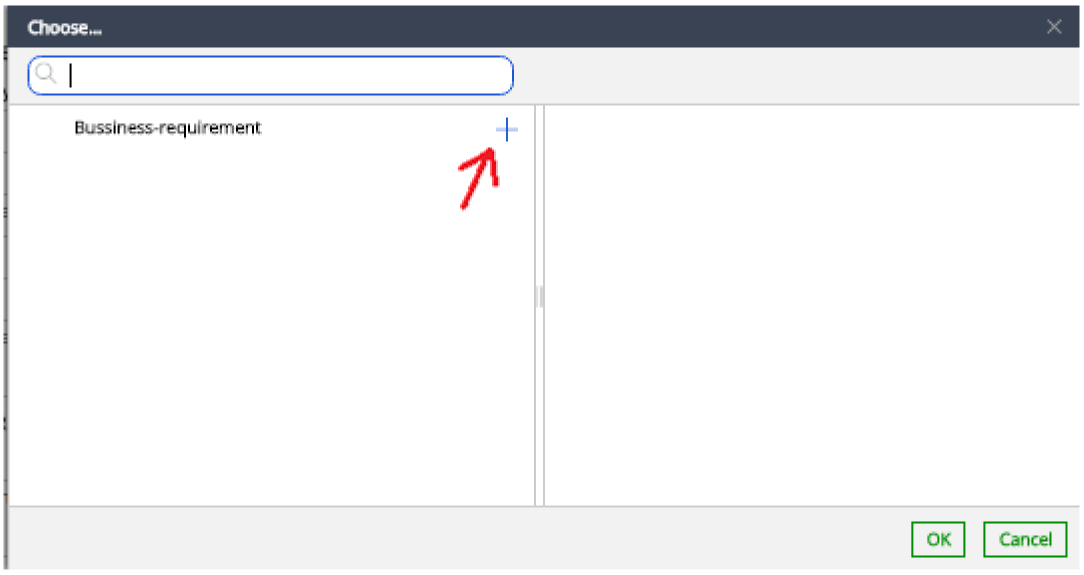
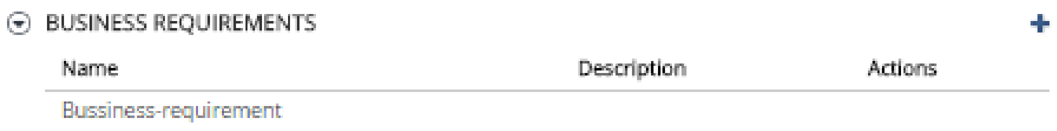


Figure 69 - Selection of business requirements

Note: If the list is empty, we enter "" into the search field and press Enter.*

After this, our requirement should appear in the "Business Requirements" cardlet (Figure 70).

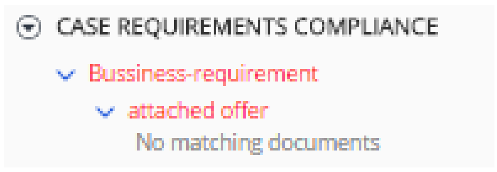


BUSINESS REQUIREMENTS

Name	Description	Actions
Bussiness-requirement		

Figure 70 - Added requirement

We return to the "Card" tab and in the right column in the "Case Compliance" cardlet we can see the following:



CASE REQUIREMENTS COMPLIANCE

Bussiness-requirement

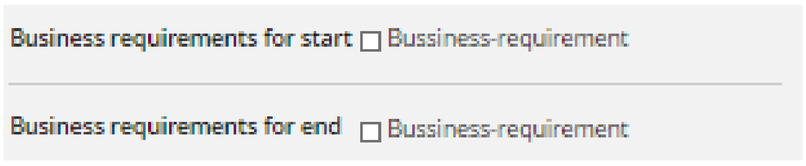
attached offer

No matching documents

Figure 71 - Compliance of the case with the requirements

As we can see, our requirement is highlighted in red, which indicates it is not fulfilled.

Now we can select the added requirement as mandatory for the task or stage to be started or ended. We go to TestStage editor and we can see that now we can select business requirements:



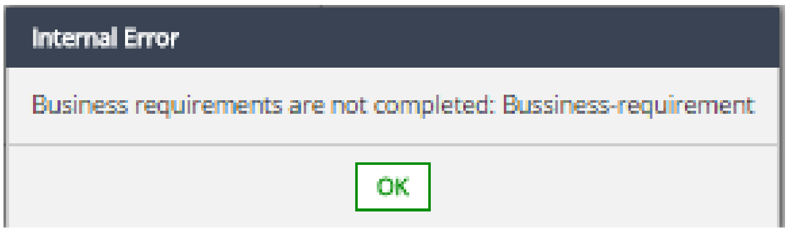
Business requirements for start ☐ Bussiness-requirement

Business requirements for end ☐ Bussiness-requirement

Figure 72 - Selection of business requirements

We check the checkbox of our requirement in the "Business Requirements for Start" field and in the "Manual started" field to verify the result. Save the changes.

Now we try to start TestStage and we can see the following:


Internal Error

Business requirements are not completed: Bussiness-requirement

Figure 73 - Error "Business requirements not fulfilled"

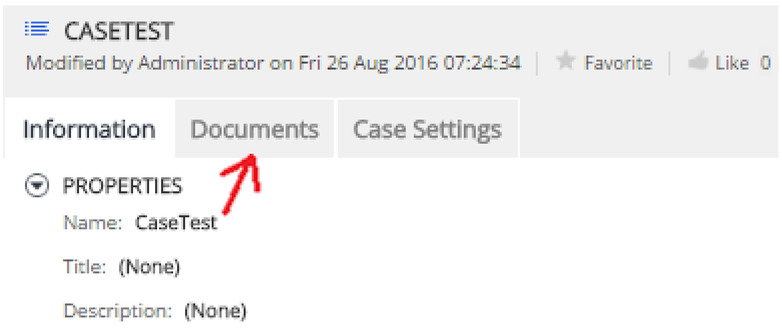
The stage did not change its status and its start will be blocked until the requirements are fulfilled.

Now we will try to start the first task in our process – «cm:title = NewTitle». Following the examples given the the previous sections, TestStage should start after completion of this task. We start it and we can see the same error. If we look at the process, we will see that none of the activities has changed its status.

In this particular case the following occurred:

The «cm:title = NewTitle» action started, then ended, then the event for TestStage start occurred, the business requirements were verified and the whole chain of changes was cancelled due to their non-fulfillment. That is, all activities returned to the state they had been in before the start button was clicked. Thus, whatever long the calls chain, all changes are cancelled if business requirements are not fulfilled at one of its stages.

Now we will try to fulfill the requirements. For this, we will go to the "Documents" tab (Figure 74).


CASETEST

Modified by Administrator on Fri 26 Aug 2016 07:24:34 | Favorite | Like 0

PROPERTIES

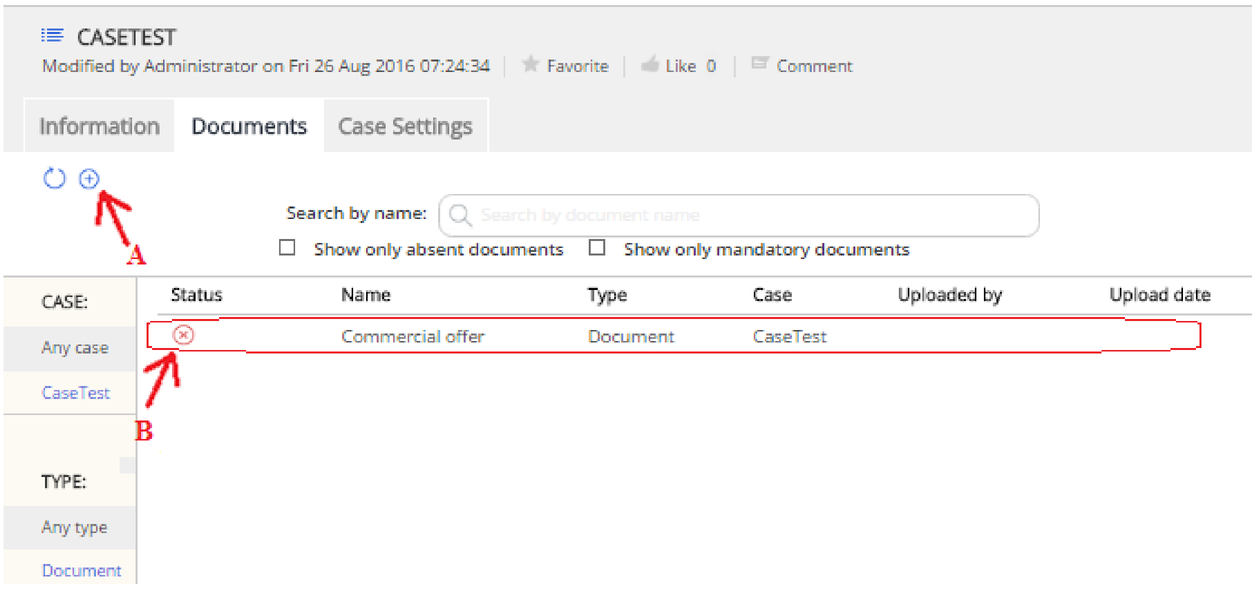
Name: CaseTest

Title: (None)

Description: (None)

Figure 74 - "Documents" tab

Here we can see that the document we added in the requirement has appeared (Figure 75).


CASETEST

Modified by Administrator on Fri 26 Aug 2016 07:24:34 | Favorite | Like 0 | Comment

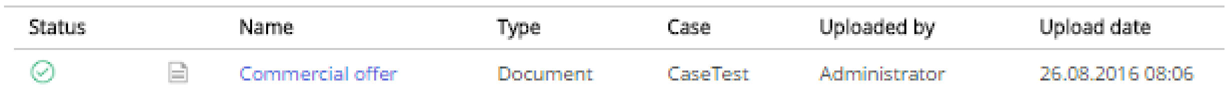
Status	Name	Type	Case	Uploaded by	Upload date
⊗	Commercial offer	Document	CaseTest		

Figure 75 - Documents

There are two ways a missing document can be uploaded:

- A - button which opens a dialog box when clicked, enabling selection of the case, type and kind of the loaded document;
- B - any file can be dragged from the computer onto the required document, and it will be automatically uploaded (drag&drop). This option is more preferable due to its simplicity.

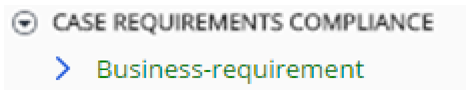
After the upload, we can see the following:



Status		Name	Type	Case	Uploaded by	Upload date
✓		Commercial offer	Document	CaseTest	Administrator	26.08.2016 08:06

Figure 76 - Result of document upload

We return to the "Card" tab and we can see that our requirement turned green in the "Case Compliance" cardlet (Figure 77).

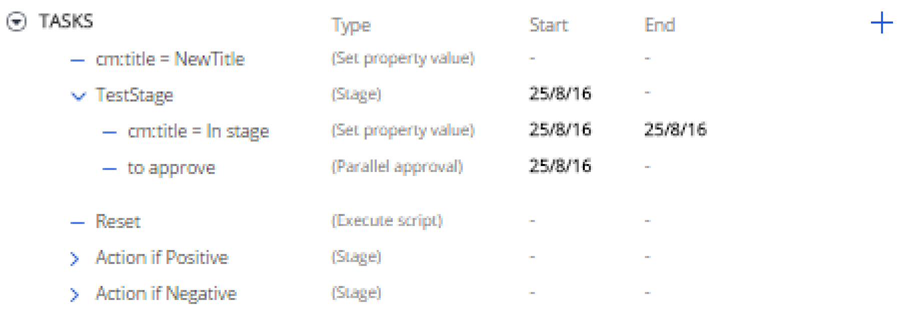


CASE REQUIREMENTS COMPLIANCE

> Business-requirement

Figure 77 - Compliance of the case with the requirements after document upload

Now we try to start TestStage:



TASKS	Type	Start	End
– cm:title = NewTitle	(Set property value)	-	-
⌄ TestStage	(Stage)	25/8/16	-
– cm:title = In stage	(Set property value)	25/8/16	25/8/16
– to approve	(Parallel approval)	25/8/16	-
– Reset	(Execute script)	-	-
> Action if Positive	(Stage)	-	-
> Action if Negative	(Stage)	-	-

Figure 78 - Result of TestStage start

The stage was successfully started since the requirements for its start had been fulfilled.

7. Case Template

We can save a created case as a template, and it will be applied to each new case. The template contains saved business requirements, tasks and roles.

To save a case as a template, click the "Save as Template" button (Figure 79).

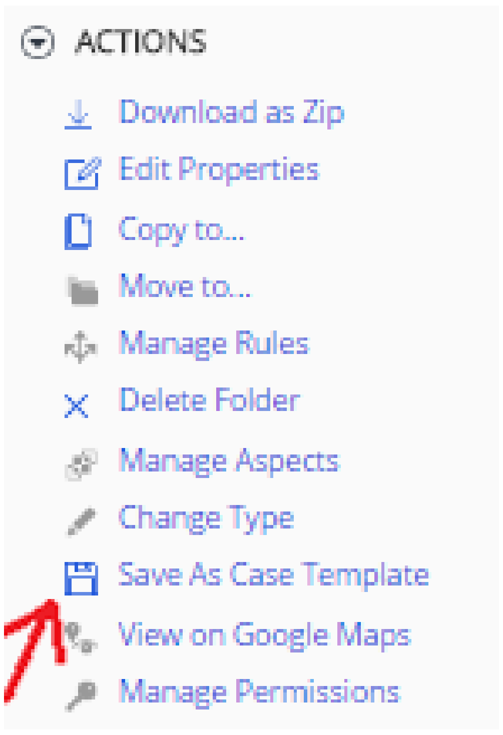


Figure 79 - "Save as Template" action

Now your process will be applied to all new cases.

Note: templates are applied to cases only («icase:case» aspect is available), i.e., if you save as a template the case we created earlier (type - cm:folder), it will not just be applied to all new folders. But if you add the «icase:case» aspect to another folder, the template will be applied to that folder. If you already have a type with icase:case among its mandatory aspects described in your model, everything will work without additional actions required.

To edit a case template, we will need to delete the old template, change a case and save it as a template.

It is convenient to use the template log for template deletion:

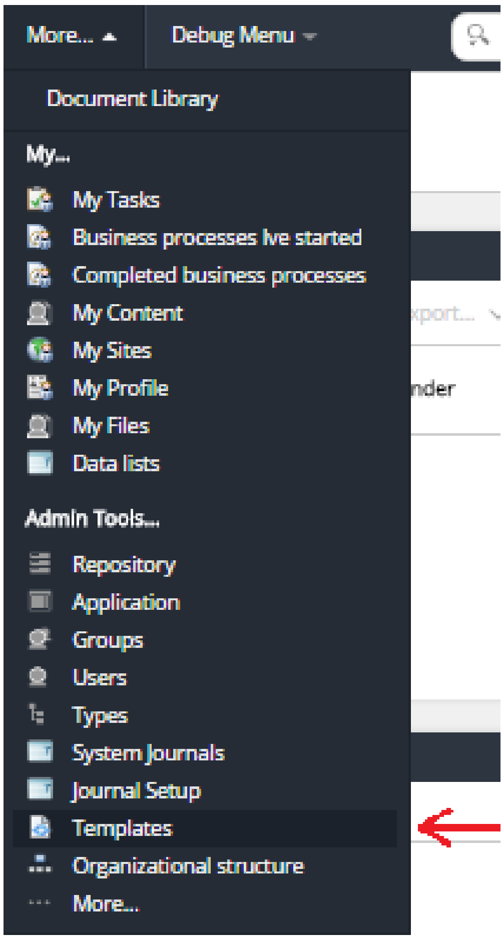


Figure 80 - Template log

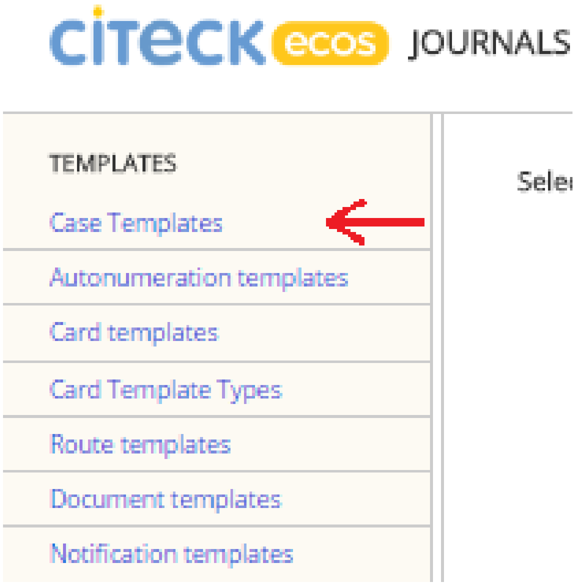


Figure 81 - Case template log

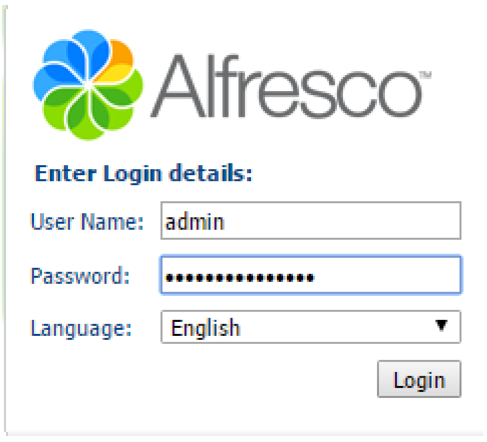
8. Case Import and Export

8.1 Export

It is desirable that we cancel all business processes linked to a case and reset the status of all activities.

At the moment, a process can be exported or imported using standard alfresco tools. For this, we need to go to the page <address>:<port>/alfresco and sign in as administrator (Figure 82).

Note: export works only for directories (cm:folder) or their children. To bypass this constraint, we can try to place the required node into a separate folder and send this construction to export.



User Name: admin
Password: •••••••••••••••
Language: English

Figure 82 - Login and password entry form

If your sign-in form did not appear, then probably you have signed in as guest. You will need to check availability of the "Logout (admin)" line in the upper right corner (Figure 83). Instead, you may see a sign-in button, and after clicking that button a login and password entry box should appear.

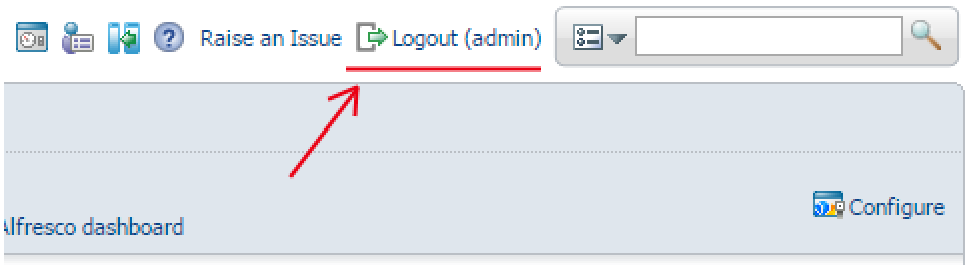


Figure 83 - Checking whether we are signed in as administrator

Next, we find a link to Company Home (Figure 84) in the browser column on the left and click on it.

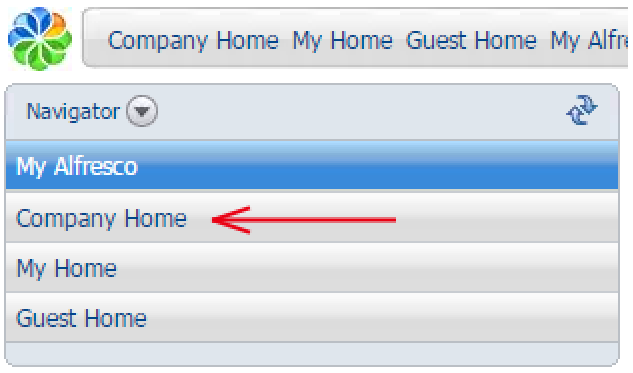


Figure 84 - Company Home

After that, we will see the contents of the repository (Figure 85).

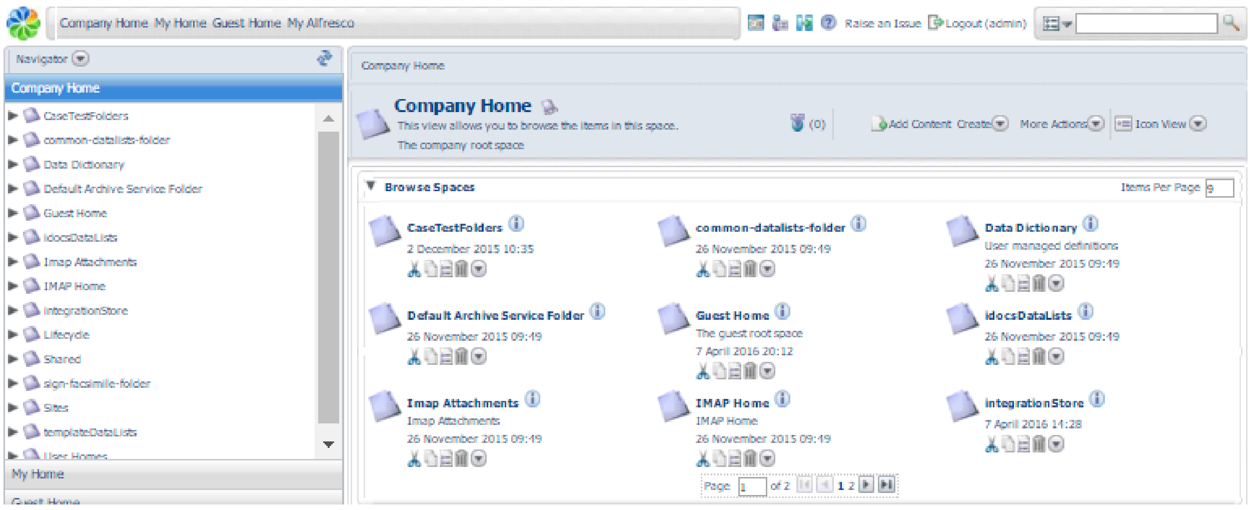
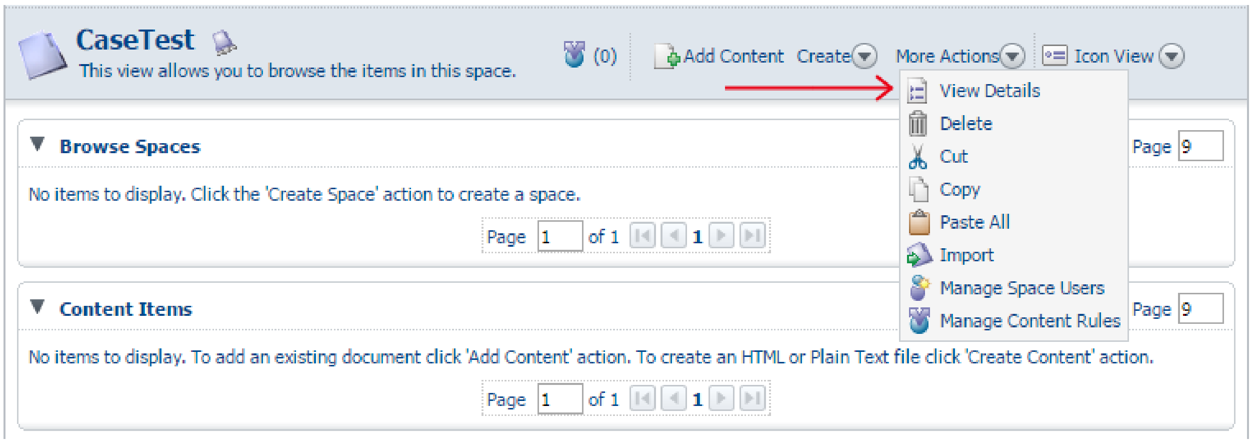


Figure 85 - Contents of the repository

We find our case, open it, open the More Actions list in the upper right corner and select View Details (Figure 86).



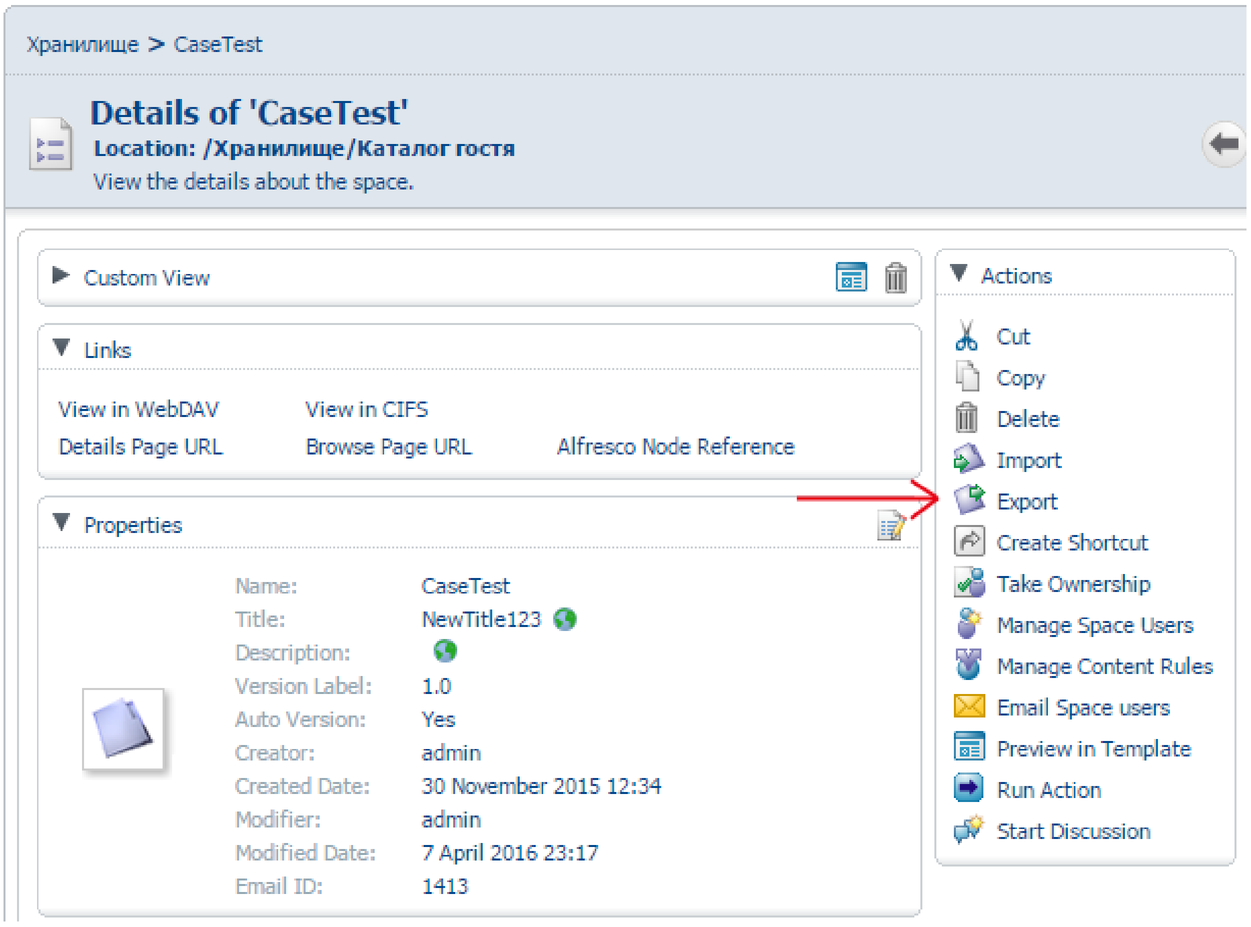
This view allows you to browse the items in this space.

No items to display. Click the 'Create Space' action to create a space.

No items to display. To add an existing document click 'Add Content' action. To create an HTML or Plain Text file click 'Create Content' action.

Figure 86 - View Details

On this page the "Export" action is of interest to us (Figure 87).



Details of 'CaseTest'

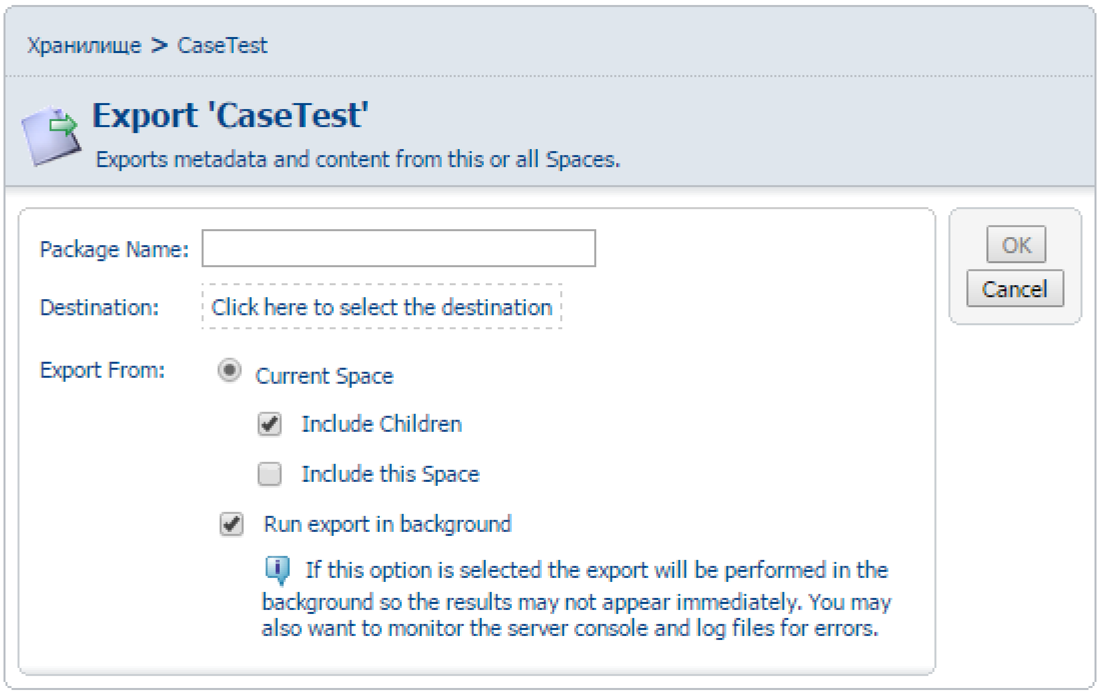
Location: /Хранилище/Каталог гостя

View the details about the space.

Name:	CaseTest
Title:	NewTitle123
Description:	
Version Label:	1.0
Auto Version:	Yes
Creator:	admin
Created Date:	30 November 2015 12:34
Modifier:	admin
Modified Date:	7 April 2016 23:17
Email ID:	1413

Figure 87 - "Export" action

When selecting the "Export" action, we will have a form open. The form is presented in Figure 88.


Хранилище > CaseTest

Export 'CaseTest'

Exports metadata and content from this or all Spaces.

Package Name:	
Destination:	Click here to select the destination
Export From:	● Current Space
	✓ Include Children
	☐ Include this Space
	✓ Run export in background

If this option is selected the export will be performed in the background so the results may not appear immediately. You may also want to monitor the server console and log files for errors.

Figure 88 - Export form

Package Name – we enter an arbitrary name for the file where the case is to be exported.

Destination – we select a directory in the repository to send our exported data. By default the current case is suggested as the target directory, but it is better to specify a different location. To do that, click the blue up arrow (Figure 89) to go up the folder hierarchy.

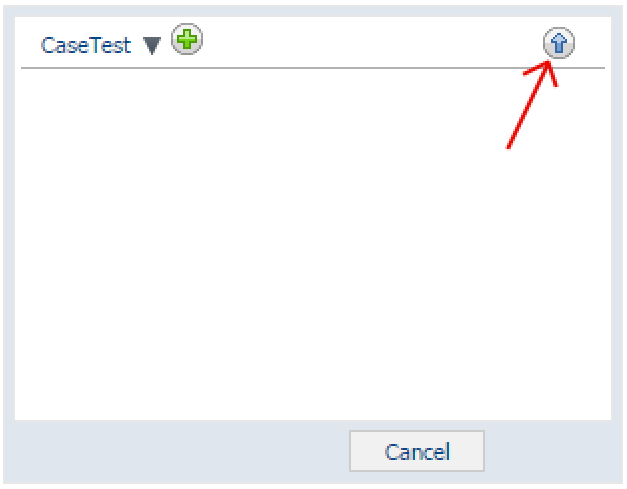


Figure 89 - Selection of the destination directory

To complete selection of the destination directory, we have to click on the green "+" opposite the directory name (Figure 90).

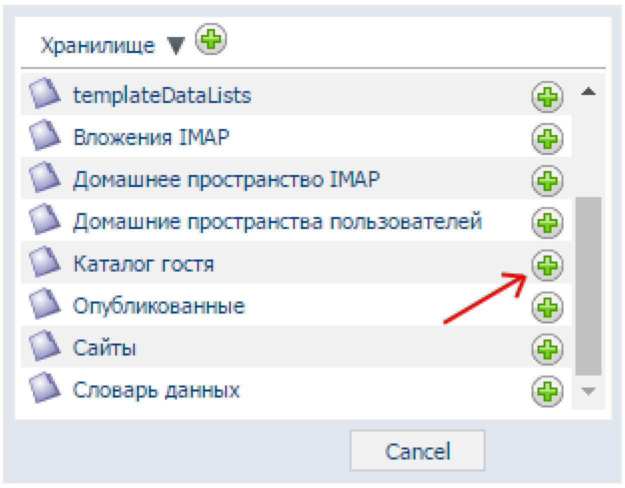
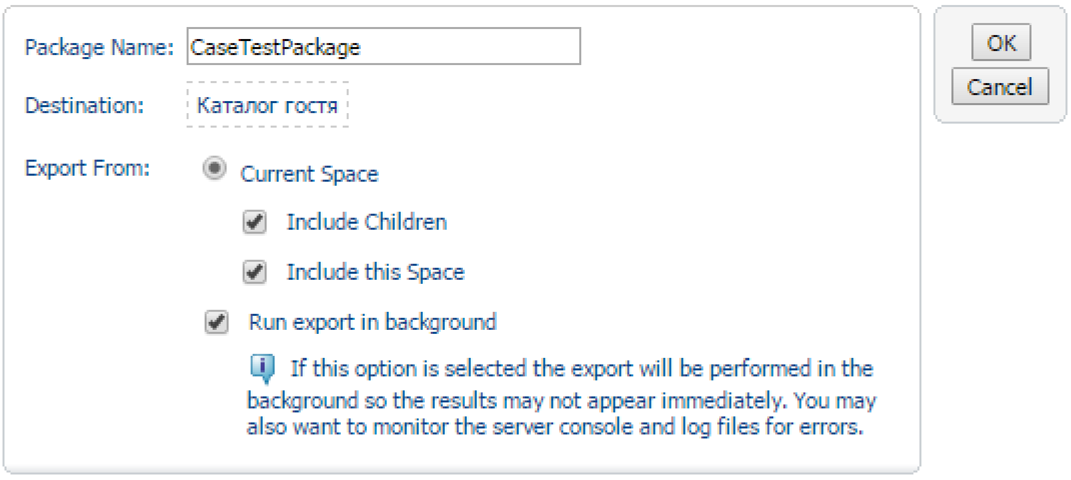


Figure 90 - Selection of the destination directory

Include Children and **Include this Space** – we make sure these checkboxes are mandatorily checked.

Figure 91 presents an example of a correctly completed export form.



Package Name: CaseTestPackage

Destination: Каталог гостя

Export From: ● Current Space

☑ Include Children

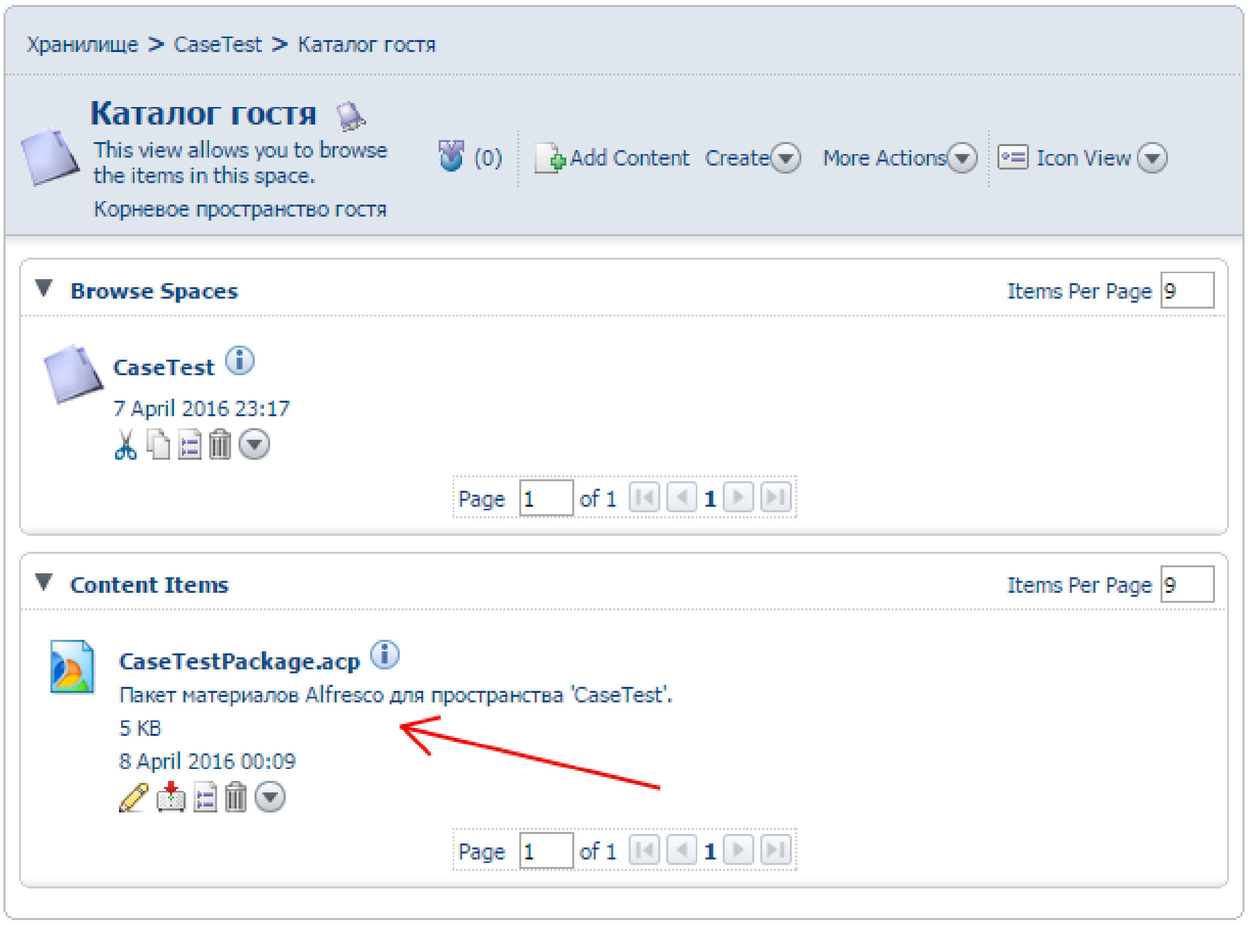
☑ Include this Space

☑ Run export in background

If this option is selected the export will be performed in the background so the results may not appear immediately. You may also want to monitor the server console and log files for errors.

Figure 91 - Example of a correctly completed export form.

Click "OK". After that our case is exported. We open the destination directory we selected during export through share/page/repository or staying in the current interface, and we can see our exported file (Figure 92).


Хранилище > CaseTest > Каталог гостя

Каталог гостя

This view allows you to browse the items in this space.

Корневое пространство гостя

(0) Add Content Create More Actions Icon View

Browse Spaces — Items Per Page 9

CaseTest
7 April 2016 23:17

Page 1 of 1

Content Items — Items Per Page 9

CaseTestPackage.acp
Пакет материалов Alfresco для пространства 'CaseTest'.
5 KB
8 April 2016 00:09

Page 1 of 1

Figure 92 - Exported case

Exported packages have an "acp" extension. We click on the *.acp file name and save it on our computer. Now we have finished exporting our case.

8.2 Import

To import a case we go to <address>:<port>/alfresco. We go to Company Home and find there the folder where we want to import our case. In the upper right corner we open More Actions and select Import (Figure 93).

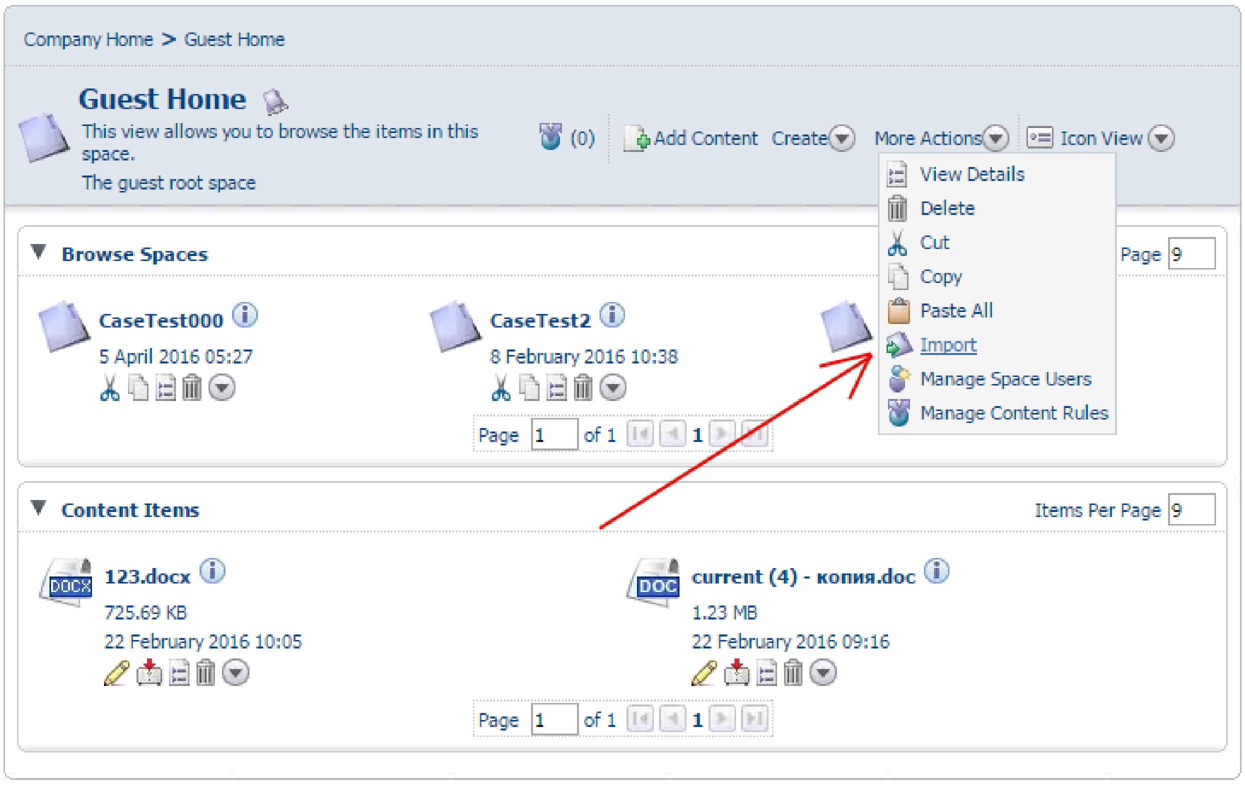
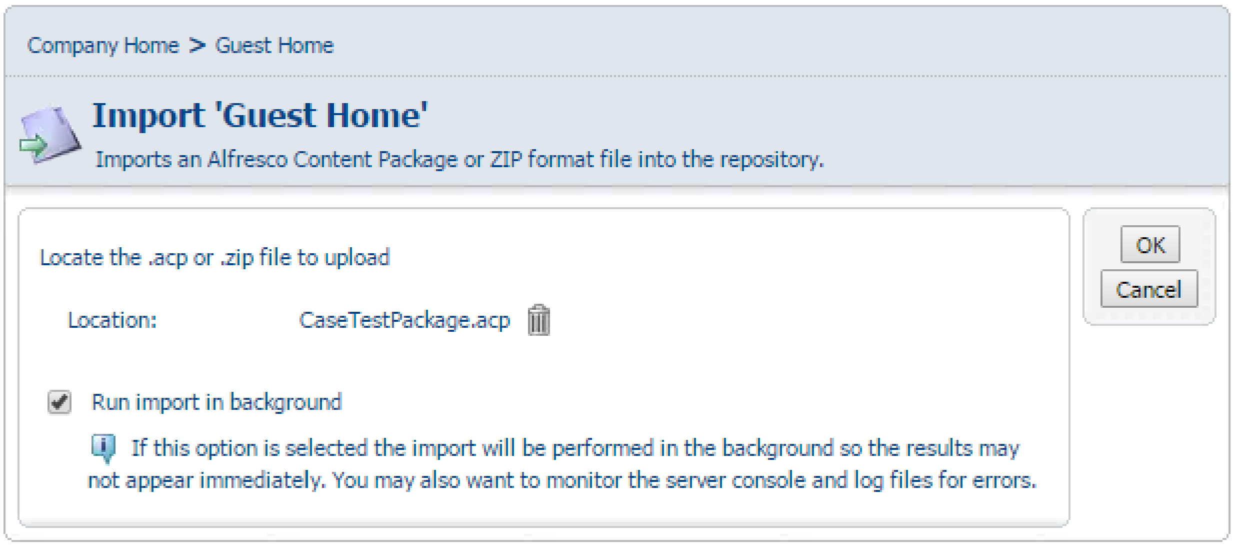


Figure 93 - "Import" action

In the opening form, we select our acp file (Figure 94).


Company Home > Guest Home

Import 'Guest Home'

Imports an Alfresco Content Package or ZIP format file into the repository.

Locate the .acp or .zip file to upload

Location: CaseTestPackage.acp

[✓] Run import in background

If this option is selected the import will be performed in the background so the results may not appear immediately. You may also want to monitor the server console and log files for errors.

Figure 94 - Case import

Click "OK". If we exit and then go back to the current folder, we can see two new objects, namely: the imported case and the acp package we selected during import (Figure 95).

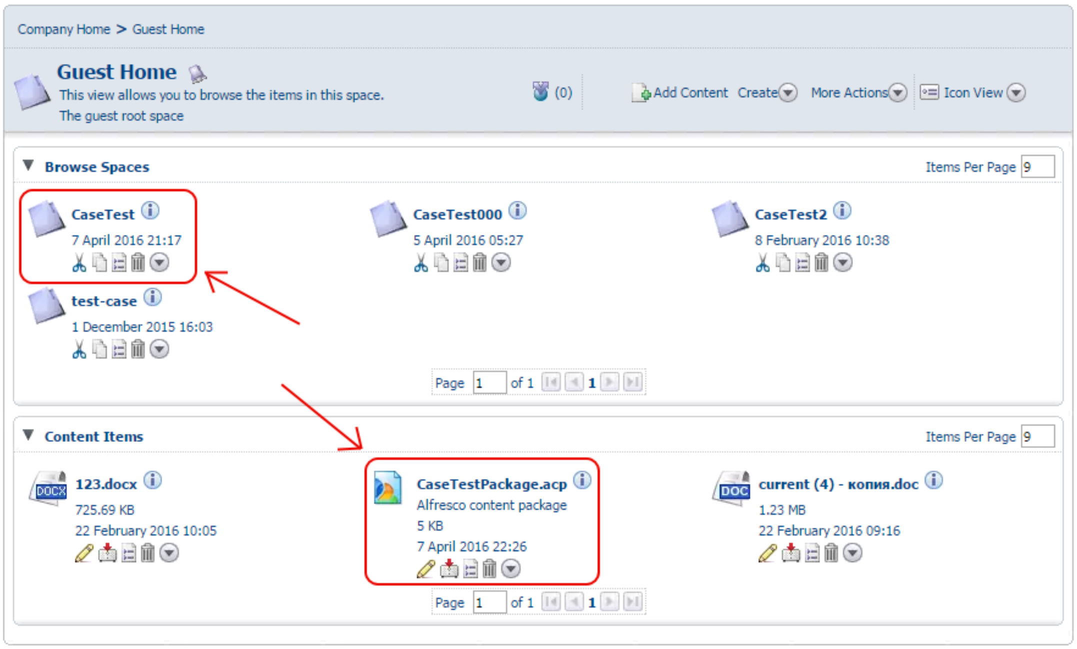


Figure 95 - Import result

Attention: When importing a complex process it may often be the case that event associations are swapped. For instance, if an event linked to the previous stage was created in a stage, then after import that event may turn out to be linked to the current stage instead of the previous one.

Attention: Mandatory associations with objects absent in the new server may cause errors during import.

Solution 1:

- We change the model in the server by removing the mandatory character of association (mandatory = false)
- Import the case
- Save the case as a template
- Delete the case
- Restore the mandatory character of association in the model (mandatory = true)

Solution 2:

- We can thoroughly examine the case during export and remove extra objects that may cause errors (e.g. sub-cases).

9. Advanced Topics

9.1 Extending the list of available tasks in a case

If the functionality of the standard types of tasks is insufficient, a new task can be added based on any BPMN process. For this, at least six stages need to be executed:

1. Creation of a BPMN process;
2. Creation of type models for tasks in the BPMN process;
3. Creation of a model for a case management task type;
4. Creation of forms for task types in the BPMN process;
5. Creation of a form for a case management task type;
6. Addition of parameter mapping from a case management task into the BPMN process;

BPMN process

By using Activity Designer, we create a simple process with a single input, single output and a single task.

Note: The process may be complex as much as desired but it is recommended to create small processes, which can be transformed to complex ones at the case management level.

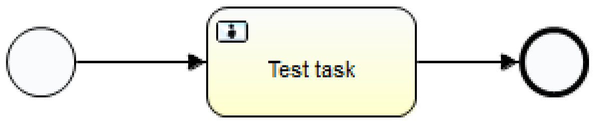
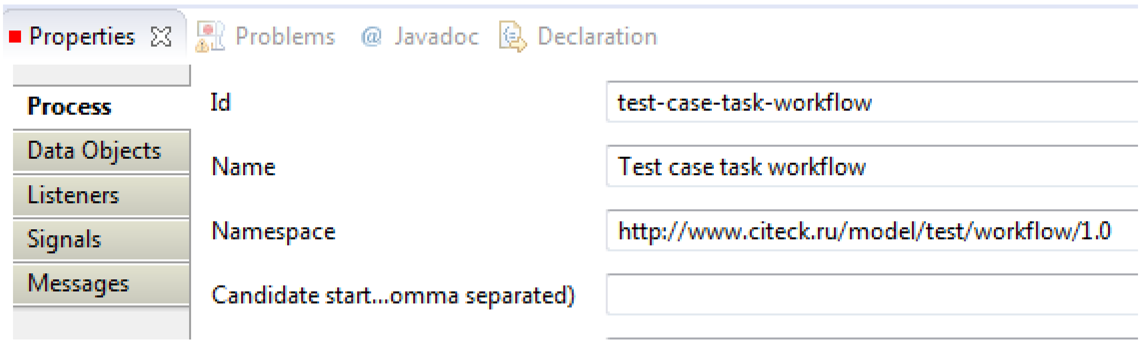


Figure 96 - Simple BPMN process

We click on an empty space in the field and set the process in the properties as shown in Figure 97.



Id: test-case-task-workflow
Name: Test case task workflow
Namespace: http://www.citeck.ru/model/test/workflow/1.0
Candidate start...omma separated)

Figure 97 - BPMN process settings

Then we highlight «Test task» and in the properties in the «Main config» tab we write **twf:testBPMNTask** in the «Form key» field.

Then we need to select a task performer. To enable selection of both a group and an individual, Citeck EcoS has an AssignTaskToAuthority class provisioned. To use it, in the task properties we go to the Linsteners tab, click the New button on the right, select

Event: create,
Type: Java class,
Class: ru.citeck.ecos.workflow.listeners.AssignTaskToAuthority
Field name: authority
Expression: ${twf_performer}

If we have difficulty selecting the class, we can open the process in the text editor and write manually:

```
<userTask id="TestTask" name="Test task" activiti:formKey="twf:testBPMNTask">
    <extensionElements>
        <activiti:taskListener event="create"
              class="ru.citeck.ecos.workflow.listeners.AssignTaskToAuthority">
            <activiti:field name="authority">
                <activiti:expression>
                    <![CDATA[${twf_performer}]]>
                </activiti:expression>
            </activiti:field>
        </activiti:taskListener>
    </extensionElements>
</userTask>
```

Models

We will create a model for our BPMN task. Generally, a type model looks as follows:

```
<type name="Type Name">
    <parent>Parent</parent>
    <properties>
        <!—Type properties -->
    </properties>
    <associations>
        <!—Type associations -->
    </associations>
    <overrides>
        <!-- Overrides properties of the parent type -->
    </overrides>
</type>
```

We will select **twf:testBPMNTask** as the name for a BPMN task type (same as during creation of the BPMN process). For user tasks in BPMN processes the parent type as a rule is**bpm:activitiOutcomeTask**. The properties of our type should contain at least one element, a property for saving the task completion results. We will name this property **twf:completeOutcome**. Since we cannot have an infinite number of task completion options, we need to impose a constraint on possible values of this property (possible options list). As a result, we have the following:

```
<property name="twf:completeOutcome">
    <type>d:text</type>
    <default>Complete</default>
    <constraints>
        <constraint name="twf:completeOutcomeOptions" type="LIST">
            <parameter name="allowedValues">
                <list>
                    <value>Complete</value>
                    <value>Reject</value>
                    <value>Other</value>
                </list>
            </parameter>
        </constraint>
    </constraints>
</property>
```

To simulate some data that need to be entered in the course of our task, we will add two fields: a string field and a Boolean field:

```
<property name="twf:someTestField">
    <title>Some text field</title>
    <type>d:text</type>
</property>
<property name="twf:someBooleanField">
    <title>Some Boolean field</title>
    <type>d:boolean</type>
</property>
```

In the associations, we need a task performer of the **cm:authority** type (this type is inherited by **cm:person** and **cm:authorityContainer**):

```
<association name="twf:performer">
    <source>
        <mandatory>false</mandatory>
        <many>true</many>
    </source>
    <target>
        <class>cm:authority</class>
        <mandatory>false</mandatory>
        <many>false</many>
    </target>
</association>
```

In the **overrides** section we need to redefine the **bpm:outcomePropertyName** property, in order to specify the name of the property where the task completion results will be stored (here we need to specify the full name of the property, where the **twf** prefix is defined as **http://www.citeck.ru/model/test/workflow/1.0**):

```
<property name="bpm:outcomePropertyName">

<default>{http://www.citeck.ru/model/test/workflow/1.0}completeOutcome</default>
</property>
```

As a result, we have the following type for our task in the BPMN process:

```
<type name="twf:testBPMNTask">
    <title>Test BPMN Task</title>
    <parent>bpm:activitiOutcomeTask</parent>
    <properties>
        <property name="twf:someTestField">
            <title>Some text field</title>
            <type>d:text</type>
        </property>
        <property name="twf:someBooleanField">
            <title>Some Boolean field</title>
            <type>d:boolean</type>
        </property>
        <property name="twf:completeOutcome">
            <type>d:text</type>
            <default>Complete</default>
            <constraints>
                <constraint name="twf:completeOutcomeOptions" type="LIST">
                    <parameter name="allowedValues">
                        <list>
                            <value>Complete</value>
                            <value>Reject</value>
                            <value>Other</value>
                        </list>
                    </parameter>
                </constraint>
            </constraints>
        </property>
    </properties>
    <associations>
        <association name="twf:performer">
            <source>
                <mandatory>false</mandatory>
                <many>true</many>
            </source>
            <target>
                <class>cm:authority</class>
                <mandatory>false</mandatory>
                <many>false</many>
            </target>
        </association>
    </associations>
    <overrides>
        <property name="bpm:outcomePropertyName">
<default>{http://www.citeck.ru/model/test/workflow/1.0}completeOutcome</default>
        </property>
    </overrides>
</type>
```

For case management tasks the model description is much simpler. If we have a single performer, then it suffices to inherit our type from **icaseTask:simpleTask** and redefine the property, which stores the name of the BPMN process to start (**icaseTask:workflowDefinitionName**):

```
<type name="tct:testCaseTask">
    <title>Test case task</title>
    <parent>icaseTask:simpleTask</parent>
    <overrides>
        <property name="icaseTask:workflowDefinitionName">
            <default>activiti$test-case-task-workflow</default>
```

```
         </property>
    </overrides>
</type>
```

We create a testModel.xml file and write there our types and additional information:

```
<?xml version="1.0" encoding="UTF-8"?>
<model name="twf:testCaseTaskModel"
       xmlns="http://www.alfresco.org/model/dictionary/1.0">

  <imports>
    <import uri="http://www.alfresco.org/model/dictionary/1.0" prefix="d" />
    <import uri="http://www.alfresco.org/model/content/1.0" prefix="cm" />
    <import uri="http://www.citeck.ru/model/icaseTask/1.0" prefix="icaseTask"/>
    <import uri="http://www.alfresco.org/model/bpm/1.0" prefix="bpm"/>
  </imports>

  <namespaces>
    <namespace uri="http://www.citeck.ru/model/test/workflow/1.0" prefix="twf" />
    <namespace uri="http://www.citeck.ru/model/test/case-task/1.0" prefix="tct" />
  </namespaces>

  <types>
      <type name="twf:testBPMNTask">
          <!-- Place a description of BPMN process tasks -->
      </type>

      <type name="tct:testCaseTask">
          <!-- Place a description of case management tasks -->
      </type>
  </types>

</model>
```

Bootstrap

In order for our new process and models to be uploaded into the system, we need to use the bootstrap functionality. For this, we create a test-context.xml file with the following content:

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE        beans        PUBLIC        '-//SPRING//DTD        BEAN//EN'
'http://www.springframework.org/dtd/spring-beans.dtd'>

<beans>

  <bean parent="workflowDeployer">
    <property name="workflowDefinitions">
      <list>
        <props>
          <prop key="engineId">activiti</prop>
          <prop  key="location">alfresco/module/idocs-repo/workflow/test-case-
task-workflow/test-case-task-workflow.bpmn</prop>
           <prop key="mimetype">text/xml</prop>
           <prop key="redeploy">true</prop>
         </props>
       </list>
     </property>
     <property name="models">
       <list>
         <value>alfresco/module/idocs-repo/model/testModel.xml</value>
       </list>
     </property>
  </bean>
</beans>
```

The paths to the model and to the BPMN process should correspond to the files you created.

Mapping

To transfer parameters from a case task to a BPMN process, parameter mapping needs to be created. For this, we add another bean to our context file:

```
<bean id="test.caseTaskAttributesMappingRegistrar"

class="ru.citeck.ecos.icase.activity.CaseTaskAttributesMappingRegistrar"
      init-method="init">
  <property name="caseTaskBehavior" ref="caseTaskBehavior" />
  <property name="attributesMappingByWorkflow">
    <map>
      <entry key="activiti$test-case-task-workflow">
        <map>
          <entry key="icaseTask:performer" value="twf:performer" />
        </map>
      </entry>
    </map>
  </property>
</bean>
```

Here we transfer the performer selected in the case task to the twf:performer association.

To have the context file load at system start, we need to add it to module-context.xml located at the following path:

{repo-module}/config/alfresco/module/{repo-module}/module-context.xml

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE beans PUBLIC '-//SPRING//DTD BEAN//EN'
'http://www.springframework.org/dtd/spring-beans.dtd'>

<beans>
    <import       resource="classpath:alfresco/module/idocs-repo/context/test-
context.xml" />
</beans>
```

Forms

To create a form for a BPMN task, we will create a testForms.xml file with the following path:

{share-module}/config/alfresco/site-config/form/testForms.xml

In this case, only the «site-config» folder is a mandatory destination condition since any share configuration is automatically retrieved from that folder.

The file content will be as follows:

```
<alfresco-config>

    <config evaluator="task-type" condition="twf:testBPMNTask">
        <forms>
            <form id="inline">
                <field-visibility>
                    <show id="twf:someTestField" />
                    <show id="twf:someBooleanField" />
                    <show id="bpm:comment" />
                    <show id="twf:completeOutcome"/>
                </field-visibility>
                <appearance>
                    <field id="twf:someTestField" />
                    <field id="twf:someBooleanField" />
                    <field id="bpm:comment">
                        <control
template="/org/alfresco/components/form/controls/textarea.ftl" />
                    </field>
                    <field id="twf:completeOutcome">
                        <control
template="/org/alfresco/components/form/controls/workflow/activiti-transitions.ftl" />
                    </field>
                   </appearance>
            </form>
        </forms>
    </config>

</alfresco-config>
```

Here attention needs to be paid to the id of the «**inline**» form. This is required for the form to be shown in the case card for the task performer.

To create a form for a case management task we create a test-case-task-views.xml file with the following path:

{repo-module}/config/alfresco/module/{repo-module}/views/test-case-task-views.xml

The file content will be as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<views xmlns="http://www.citeck.ru/ecos/views/1.0">
  <imports>
    <import uri="http://www.citeck.ru/model/icaseTask/1.0" prefix="icaseTask"/>
    <import uri="http://www.citeck.ru/model/test/case-task/1.0" prefix="tct" />
  </imports>

  <view class="tct:testCaseTask">
    <view class="icaseTask:task"/>
    <field assoc="icaseTask:performer"/>
  </view>
</views>
```

We start the system, open the case, click "+" in the "Tasks" cardlet and in the Task sub-menu we can see our new task (Figure 98).

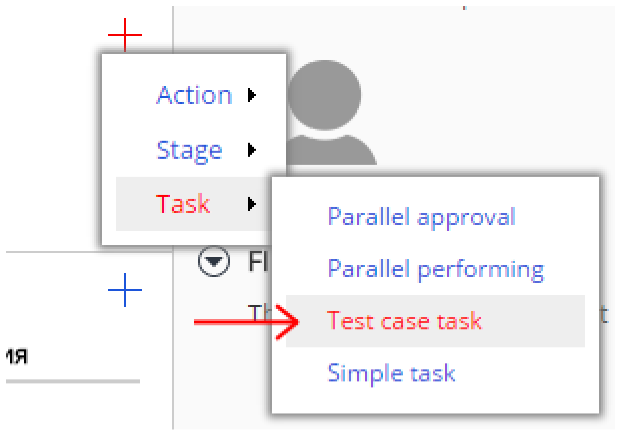
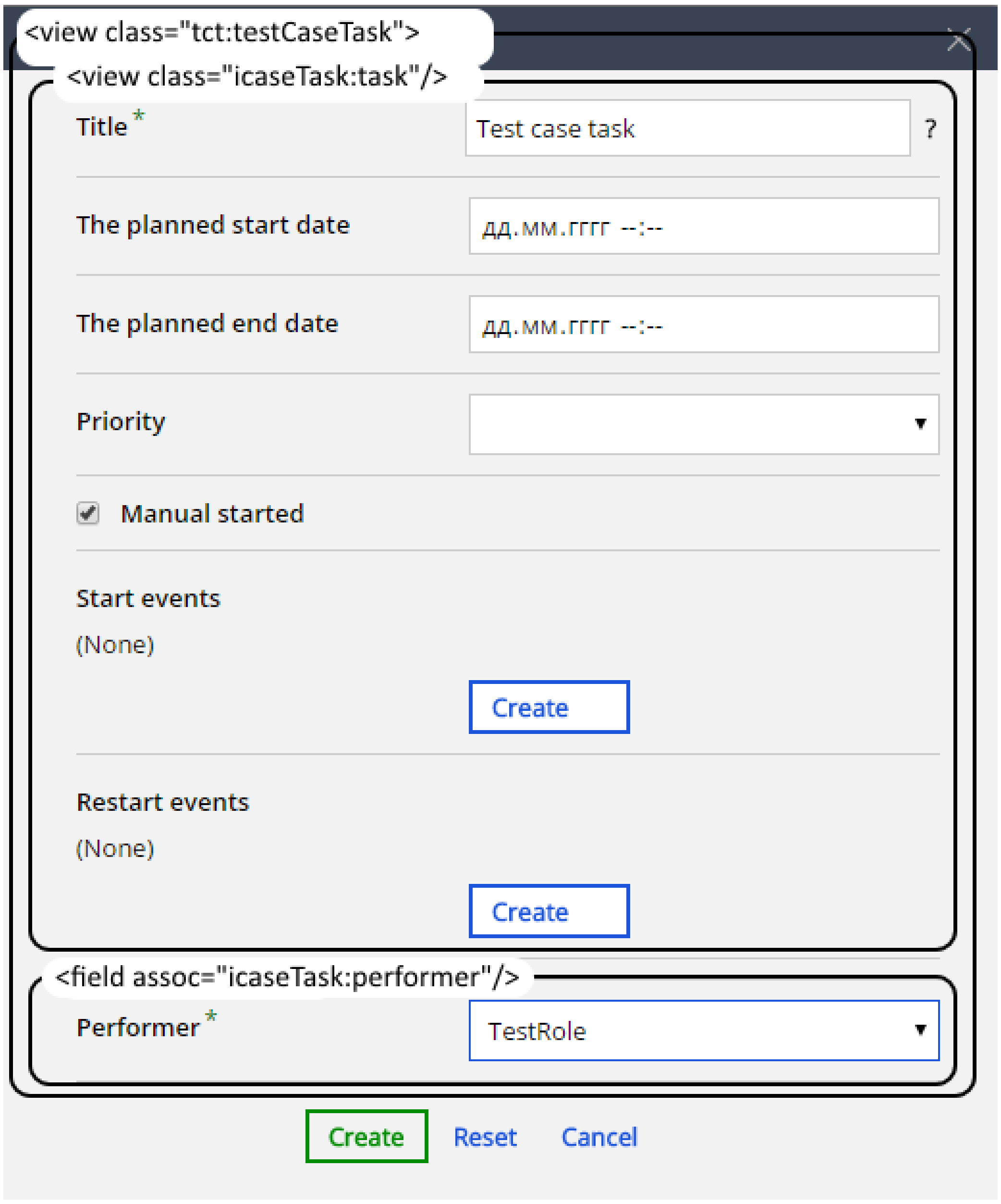


Figure 98 - Creation of a new case task

We open the creation form for this task. Figure 99 presents an open creation form and parts of the form are marked in accordance with **test-case-task-views.xml**. The upper part of the form shows fields predefined for **icaseTask:task**. The bottom part contains an additional field **icaseTask:performer**, which is required to select a task performer.


```
<view class="tct:testCaseTask">
<view class="icaseTask:task"/>
```

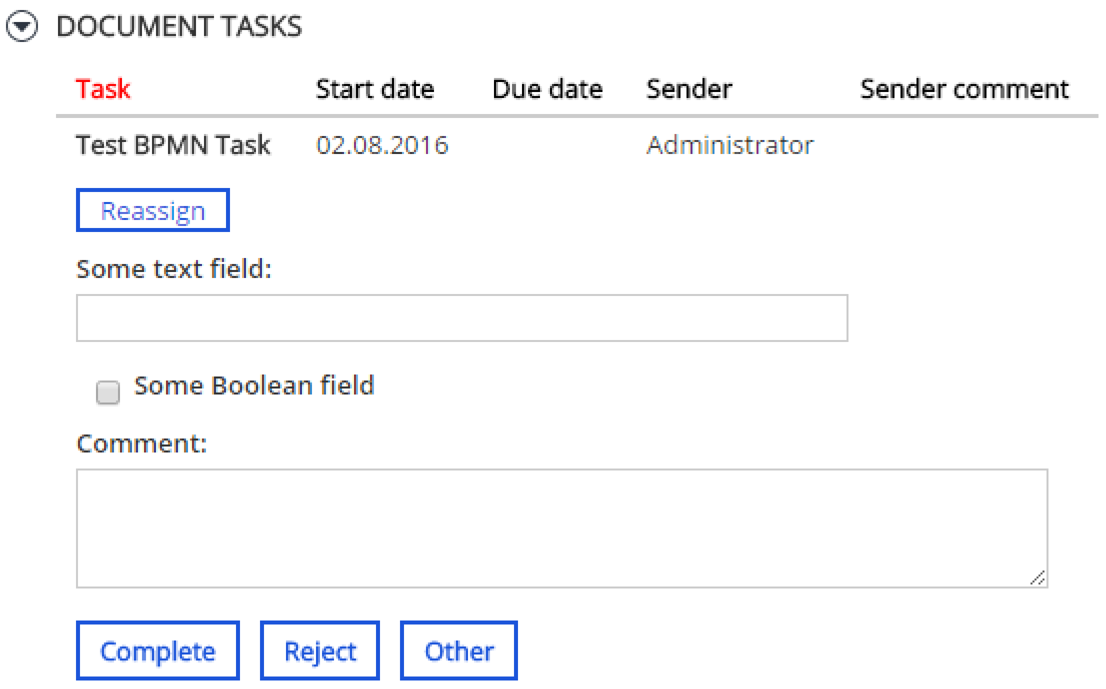
Title *	Test case task
The planned start date	дд.мм.гггг --:--
The planned end date	дд.мм.гггг --:--
Priority	
✓ Manual started	
Start events	
(None)	
Restart events	
(None)	

```
<field assoc="icaseTask:performer"/>
```

Performer *	TestRole

Figure 99 - Creation form for a task

We finish creating the new task by clicking on "Create" and start the task. In the "Document Tasks" cardlet we can see the form we described in testForms.xml (Figure 100).


DOCUMENT TASKS

Task	Start date	Due date	Sender	Sender comment
Test BPMN Task	02.08.2016		Administrator	

Reassign

Some text field:

Some Boolean field

Comment:

Figure 100 - Form of the BPMN task in the document card

At this stage creation of the new task for case management is completed.

9.2 Dynamic roles

Often role representatives have to be selected depending on certain conditions. For instance, if we have a contract case, the performer of the "signatory" role may depend on the contract amount. To realize such roles, a javascript action needs to be created which, when executed, will fill the role with needed individuals or groups. We will look at an example of creating such action.

We create one role named "TestRole". We select any group or an individual as representatives.

We create a javascript action, which will be responsible for selecting role representatives. We will dwell more on the content of the "javascript code".

Auxiliary functions:

- Retrieve role from case on name

```
function getRole(roleName) {
        var roles = document.childAssocs['icaseRole:roles'] || [];
        for (var i in roles) {
            if (roles[i].properties.title == roleName) {
                return roles[i];
            }
        }
        return null;
    }
```

- Remove role representatives

```
    function removeAssignees(role) {
        var assignees = role.assocs['icaseRole:assignees'] || [];
        for (var i in assignees) {
            role.removeAssociation(assignees[i], 'icaseRole:assignees');
        }
    }
```

- Add role representative

```
    function addAssignee(role, assignee) {
        role.createAssociation(assignee, 'icaseRole:assignees');
    }
```

Having written the functions described above we can already realize selection of the current user as a role representative:

```
(function() {

    var role = getRole('TestRole');
    removeAssignees(role);
    addAssignee(role, person);

})()
```

This scenario may be useful, e.g. when a task is assigned to a group and after the task is completed, the next task needs to be performed by the person who completed the previous one.

A second option to select role representatives is to select particular individuals or groups depending on some conditions. E.g., we can assign a person by his/her username or a group by its name (a group name always begins with the prefix «GROUP_»):

```
    (function() {

        var role = getRole('TestRole');
        removeAssignees(role);

        var assignee;
        if (Math.random() > 0.5) {
            assignee = people.getPerson("admin");
        } else {
            assignee = people.getGroup("GROUP_ALFRESCO_ADMINISTRATORS");
        }
        addAssignee(role, assignee);

    })()
```

Roles can also be selected from the orgchart. For this, the role name and the name of the parent group is required (to search the whole orgchart **GROUP__orgstruct_home_** can be used as the parent group). The following function is required to perform the search:

```
/**
* Get first group of type 'type', which is subgroup of group 'groupName'.
*
* @param groupName parent group name
* @param type group type
* @param immediate specify 'true' for only immediate subgroups, or 'false' for all
subgroups
* @return group of specified type, that is subgroup of specified group, or null
*/
public ScriptGroup getTypedSubgroup(String groupName, String type, boolean immediate);
```

Example: we will select the company Director from the default orgchart. We will have **GROUP_company** as the parent group; the director role will be **director**.

```
(function() {

    var role = getRole('TestRole');
    removeAssignees(role);

    var parent = "GROUP_company";
    var orgstructRole = orgstruct.getTypedSubgroup(parent, 'role', "director", false);
    addAssignee(role, orgstructRole.getGroupNode());

})()
```