

Workflow Expander

Stand: 28.07.2015

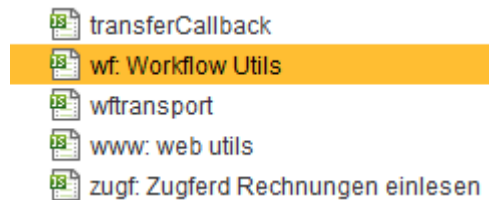
Es kommt in Workflowprozessen sehr oft vor, dass in Abhängigkeit von externen Daten unterschiedliche serielle oder parallele Freigaben oder Benachrichtigungen ausgeführt werden müssen. Dazu muss der Workflow durch eine ELOas Regel dynamisch erweitert werden. Im Folgenden werden ein paar Hilfsfunktionen vorgestellt, die solche Ergänzungen mit minimalen Aufwand ermöglichen.

Inhalt

1	Installation	2
2	Einbindung in einen Workflow	3
2.1	Lineare Workflows	5
2.2	Parallele Workflows	6
2.3	Erzeugung der Anwenderliste	8
2.3.1	Beispiel 1: die Anwenderliste steht in der Indexzeile mit dem Gruppennamen „APPROVAL“	8
2.3.2	Beispiel 2: die Anwender stehen in den Indexzeilen USER1, USER2 und USER3.....	9

1 Installation

Die Erweiterungen sind Teil der ELOas Library „wf: Workflow Utils“. Zur Installation reicht es also aus, dieses Modul zu aktualisieren. Es kann dann in neuen Regeln verwendet werden.



Es werden dann zusätzlich diese Funktionen zur Verfügung gestellt:

`wf.expandNodeParallel(workflowId, nodeId, userList)`

`wf.expandNodeLinear(workflowId, nodeId, userList)`

```
// start namespace wf
var wf = new Object();
wf = {

  expandNodeParallel: function(workflowId, nodeId, userList) {
    var flow, node, matrix, successorId, firstSuccessor, sndSuccessor,
    var nodeInserter = new NodeInserter();

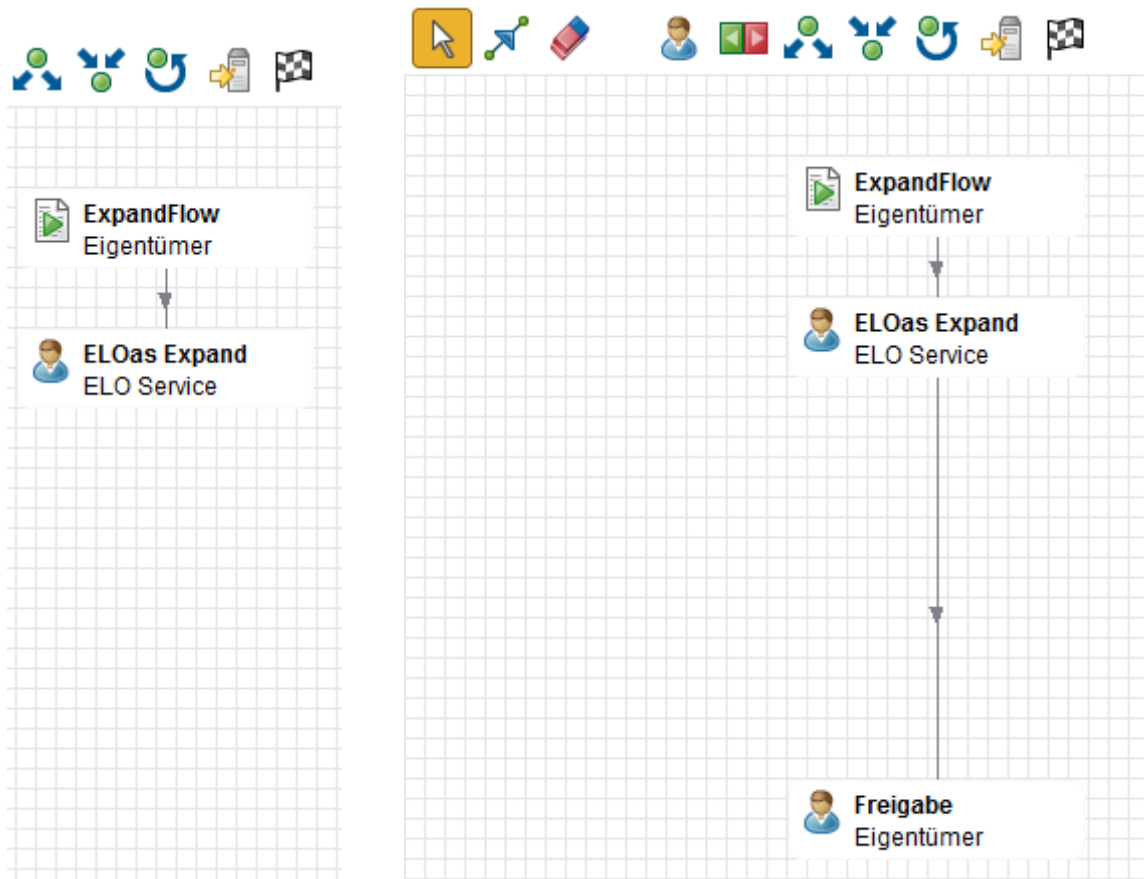
    try {
      flow = this.readWorkflow(workflowId, true);
      var nodes = nodeInserter.copyNodes(flow);
      var assocs = nodeInserter.copyAssocs(flow);
      node = wf.getNodeById(flow, nodeId);
      matrix = flow.matrix.assocs;
      for (var a = 0; a < matrix.length; a++) {
        var assoc = matrix[a];
        if (assoc.nodeFrom == nodeId) {
          if (!firstSuccessor) {
            firstSuccessor = wf.getNodeById(flow, assoc.nodeTo);
          } else if (!sndSuccessor) {
            sndSuccessor = wf.getNodeById(flow, assoc.nodeTo);
          } else {
            break;
          }
        }
      }
    }

    nodeInserter.insertNodesParallel(flow, nodes, assocs, node, firstSuccessor, sndSuccessor, userList);
    flow.nodes = nodes;
    flow.matrix.assocs = assocs;
    this.writeWorkflow(flow);
    flow = null;
  } finally {
    if (flow) {
      this.unlockWorkflow(flow);
    }
  }
}
```

2 Einbindung in einen Workflow

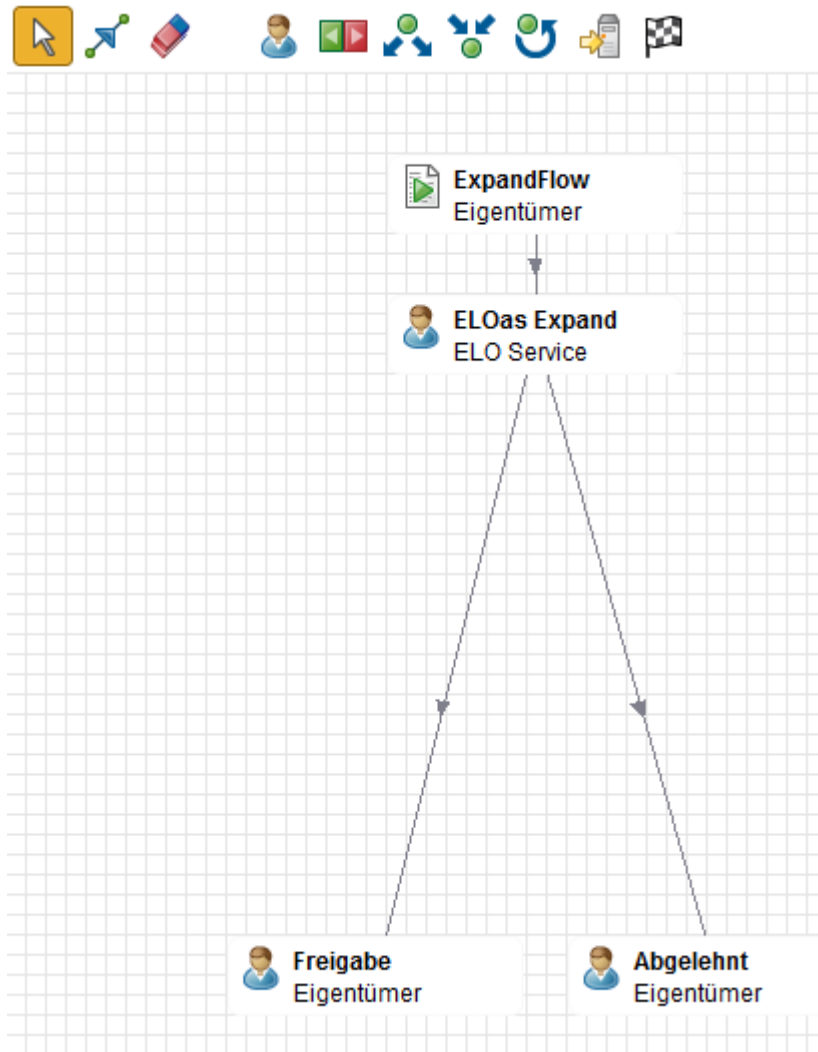
Die Workflow-Erweiterung kann über einen ELOas Knoten in einen beliebigen Workflow eingebunden werden. Der ELOas Knoten wird dann automatisch erweitert.

Wenn der ELOas Knoten keinen oder nur einen Nachfolger hat, wird eine einfache Benachrichtigung erzeugt.



Wenn es einen Nachfolger gibt, dann ist es sinnvoll, dass man im Workflowdiagramm etwas Platz zwischen den Knoten lässt. Dort werden dann die neu generierten Knoten abgelegt.

Gibt es zwei Nachfolger, dann wird der erste Nachfolger als „Freigabe“ und der zweite Nachfolger als „Abgelehnt“ angesehen. Es wird dann ein passender linearer oder paralleler Freigabeworkflow erzeugt.



Damit der „ELOas Expand“ Knoten erweitert wird, muss dieser in einen Workflow Ruleset eingebunden werden. Die nachfolgenden Beispiele gehen davon aus, dass es nur diese Workflowbearbeitung gibt. In einem echten Projekt wird es natürlich mehrere unterschiedliche Workflow Rulesets geben. Dann muss jeder Ruleset prüfen, welche Workflows für ihn bestimmt sind und er darf dann auch nur diese bearbeiten.

Der Ruleset für die Workflowerweiterung kann direkt über den Designer aus der Admin Console heraus erstellt werden. Es muss, neben der Prüfung, ob dieser Knoten überhaupt von diesem Ruleset bearbeitet werden soll, nur ein Aufruf zu der Funktion `wf.expandNodeLinear` oder `wf.expandNodeParallel` erfolgen.

Diese beiden Funktionen besitzen die gleichen Parameter. Der erste Parameter gibt die Nummer des Workflows an, der zweite Parameter die Nummer des Workflowknotens, der erweitert werden soll. Diese beiden Werte können Sie aus `EM_WF_NODE` ermitteln, dem aktuellen Workflowknoten des Aufrufs. Der Dritte Parameter enthält eine Liste mit den Namen der Anwender, die in dem Workflow ergänzt werden sollen. Diese Liste kann aus den Prozessdaten ermittelt werden, z.B. aus einem Indexfeld.

2.1 Lineare Workflows

expand

Assistent Skript

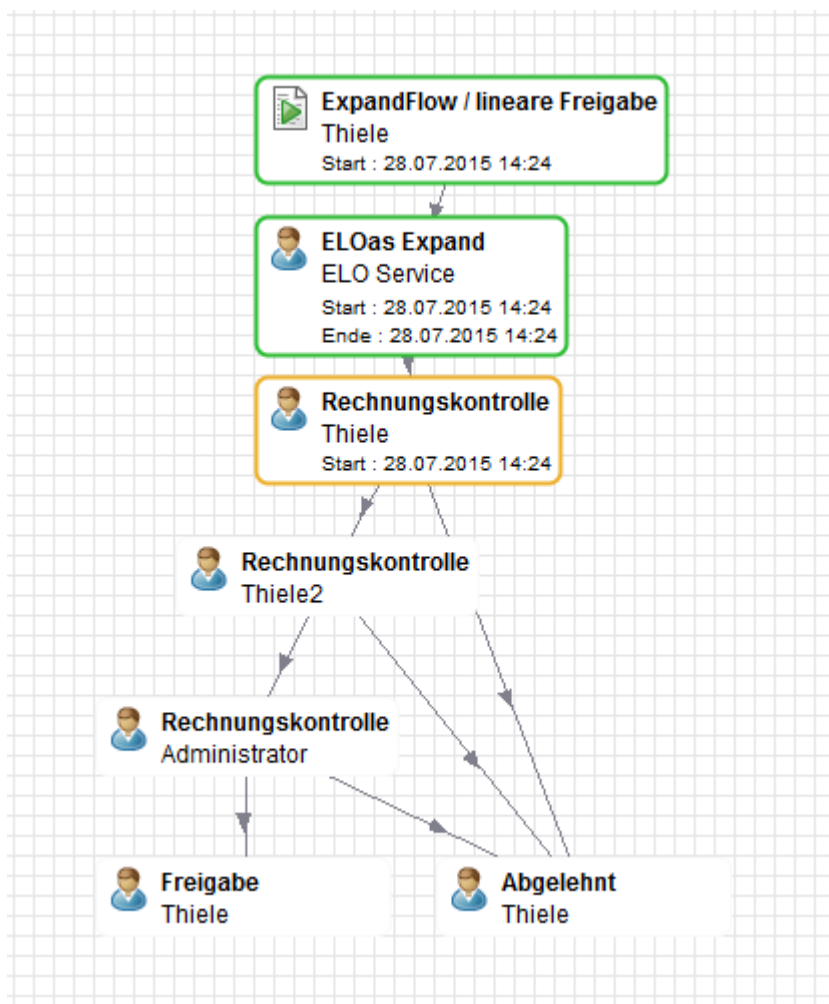
☐ JavaScript Editor benutzen

```
//Generated Rule code : expand
log.debug("Process Rule expand.");
wf.expandNodeLinear( EM_WF_NODE.flowId, EM_WF_NODE.nodeId, ["Thiele", "Thiele2", "Administrator"] );
EM_WF_NEXT = "0";
```

Ein Aufruf von

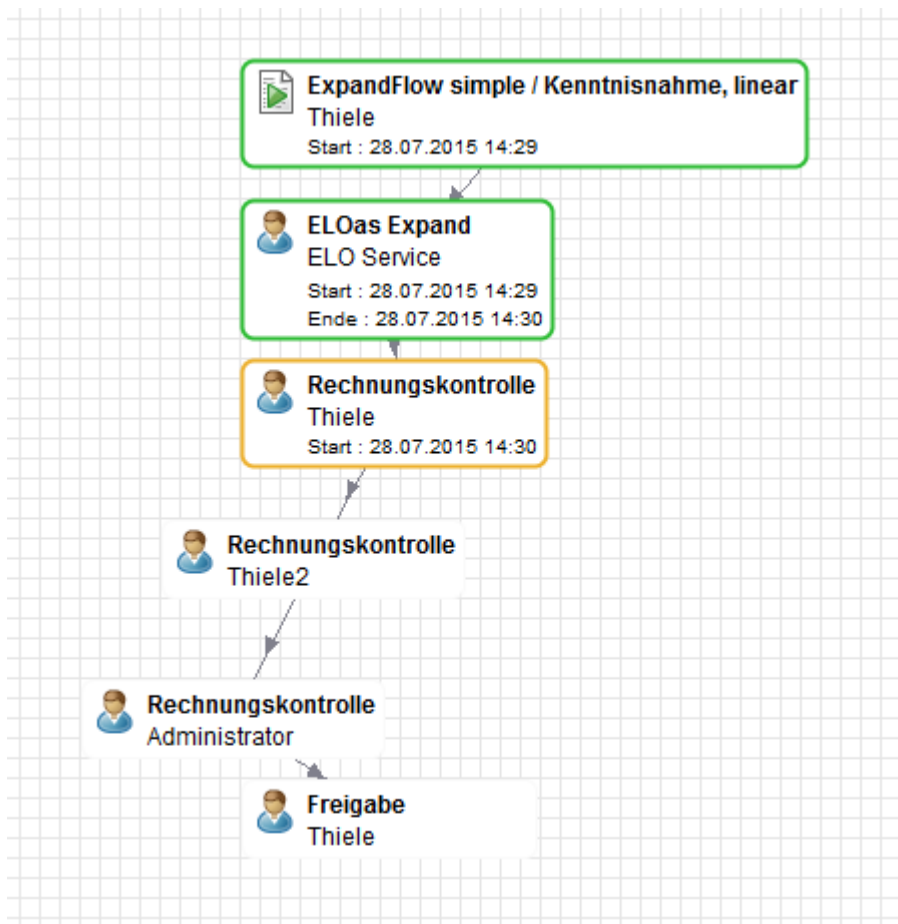
```
wf.expandNodeLinear( EM_WF_NODE.flowId, EM_WF_NODE.nodeId, ["Thiele", "Thiele2", "Administrator"], "Rechnungskontrolle" );
```

erzeugt dann diesen Freigabeworkflow:



Die drei Anwender aus der Parameterliste erhalten den Workflow nacheinander. Jeder kann die Rechnungskontrolle weiterleiten oder abbrechen.

Wenn der ELOas Knoten nur einen Nachfolger hat, wird eine einfache Kenntnissnahme erzeugt.

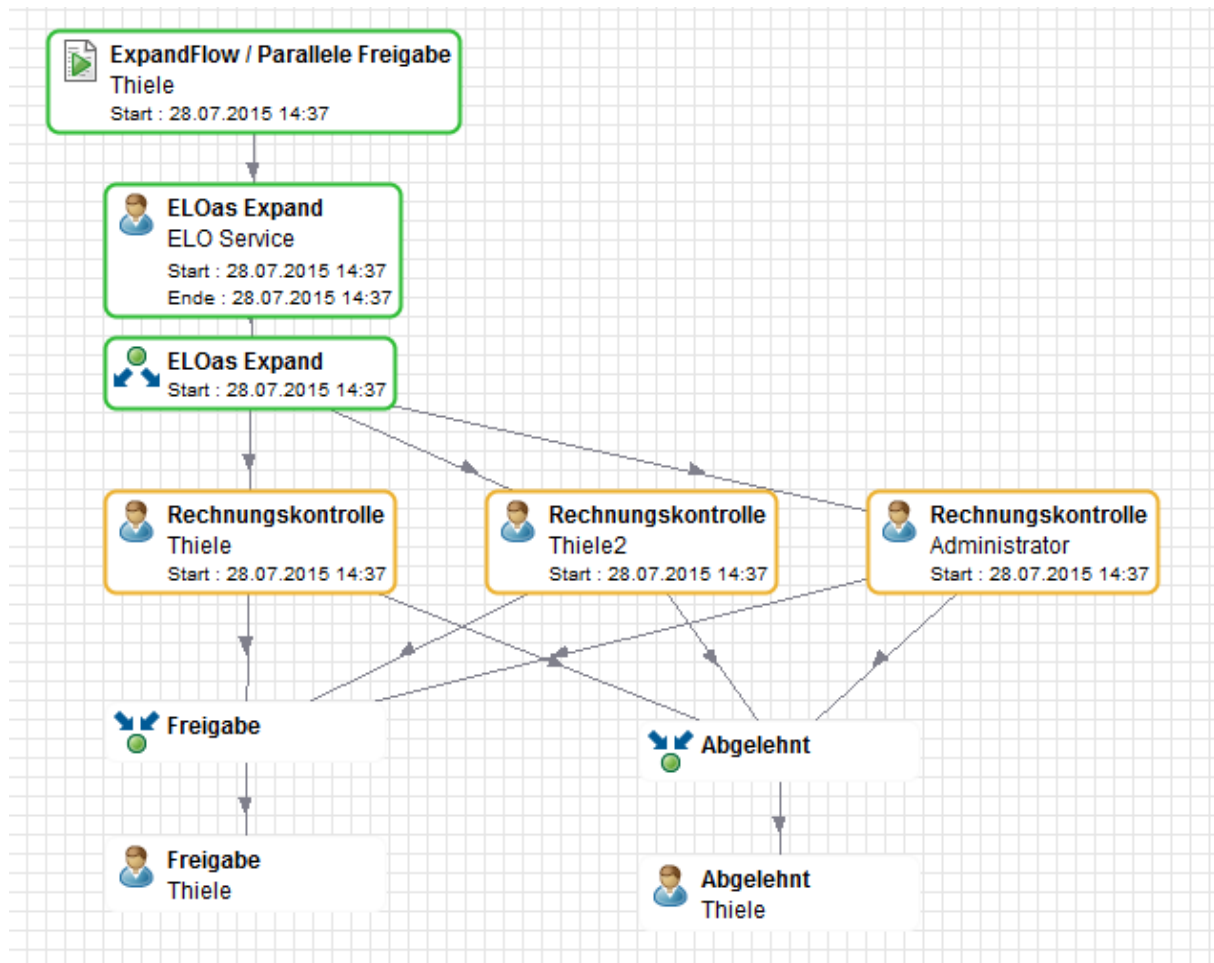


2.2 Parallele Workflows

Ein Aufruf von

```
wf.expandNodeParallel( EM_WF_NODE.flowId, EM_WF_NODE.nodeId, ["Thiele", "Thiele2",  
"Administrator"], "Rechnungskontrolle" );
```

erzeugt diesen parallelen Freigabeworkflow:




Wenn einer der Anwender die Freigabe ablehnt, ist es nicht sinnvoll, dass die Aufgabe bei den anderen Anwendern in der Aufgabenliste verbleibt. Sobald eine Ablehnung vorliegt, kann die Gesamtaufgabe nicht mehr angenommen werden. Aus diesem Grund werden mit der ersten Ablehnung alle noch offenen Aufgaben zurückgenommen. Das passiert in dem Sammelknoten „Abgelehnt“. Er wird so angelegt, dass er bereits mit dem ersten Eingang aktiv wird und er besitzt eine Liste der generierten Personenknoten, die beim Betreten zurückgesetzt werden müssen.

☐ Auf alle Vorgängerknoten warten

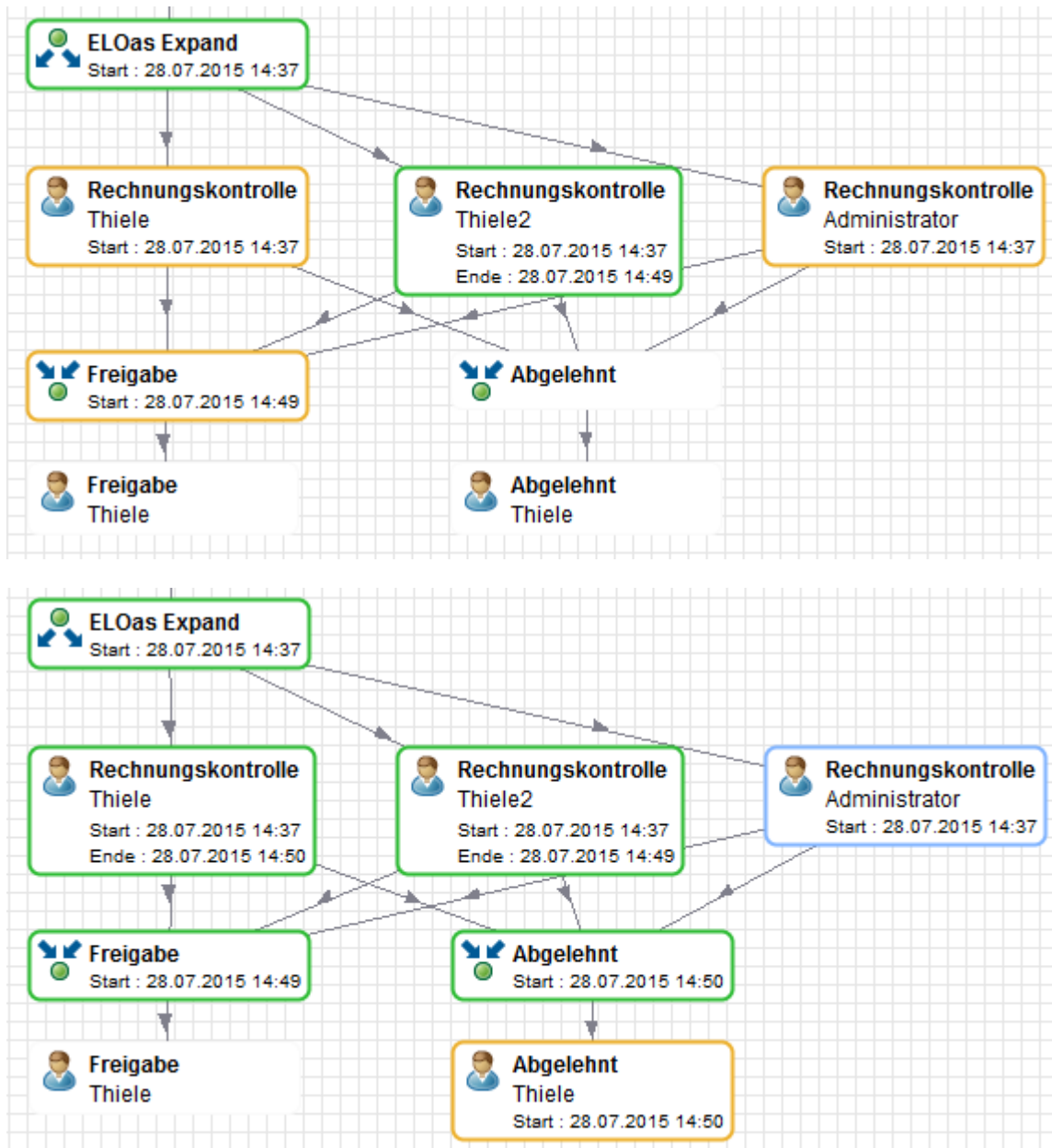
☒ Weiterleiten bei Anzahl abgeschlossener Vorgängerknoten

1

Diese Knoten beim Weiterleiten deaktivieren

7,8,9,5 

Im Folgenden wird der Workflow zuerst vom Anwender Thiele2 freigegeben, dann vom Anwender Thiele abgelehnt:



Man sieht, dass der Workflow zum Abschluss im Zustand „Abgelehnt“ ist. Obwohl der Administrator noch gar nicht reagiert hat, ist sein Knoten deaktiviert worden. Man erkennt an der blauen Umrandung, dass ihm der Knoten entzogen wurde.

2.3 Erzeugung der Anwenderliste

Die einfachen Beispiele haben die Anwenderliste statisch in einem festen Array mit Namen übergeben. Im Normalfall wird man diese Liste aber dynamisch über den Prozess erzeugen.

2.3.1 Beispiel 1: die Anwenderliste steht in der Indexzeile mit dem Gruppennamen „APPROVAL“

Die Namen sind jeweils durch ein Semikolon getrennt, das Array wird über einen regulären Ausdruck erzeugt:

APPROVAL.split(/;/g)

```
//Generated Rule code : expand
log.debug("Process Rule expand.");
wf.expandNodeParallel( EM_WF_NODE.flowId, EM_WF_NODE.nodeId, APPROVAL.split(/;/g), "Rechnungskontrolle" );
EM_WF_NEXT = "0";
```

2.3.2 Beispiel 2: die Anwender stehen in den Indexzeilen USER1, USER2 und USER3

Das Array wird durch eine Zusammenfassung der drei Variablen zu einem Array erzeugt:

wf.expandNodeParallel(flowId, nodeId, [USER1, USER2, USER3], "Kontrolle");

```
//Generated Rule code : expand
log.debug("Process Rule expand.");
wf.expandNodeParallel( EM_WF_NODE.flowId, EM_WF_NODE.nodeId, [USER1, USER2, USER3], "Rechnungskontrolle" );
EM_WF_NEXT = "0";
```

Revisionsgeschichte dieses Dokuments			
Version	Datum	Bearbeiter	Änderungen
1.0	28.07.2015	Thiele	