На платформе Alfresco ECM

Процессы в Citeck EcoS

2015

Оглавление

Введение

Основное назначение процессов в Citeck EcoS – это обеспечение следования документа своему жизненному циклу. То есть, у определенного типа документа (например, у Входящего) может быть свой жизненный цикл, начиная с момента попадания его в систему до, например, перемещения его в архив, когда по документом уже не проводится никаких активных действий.

Инструментов для создания процессов в Citeck EcoS используется два:

1. Процессы Activiti (http://activiti.org/);
2. Жизненный цикл документа Citeck EcoS (внутренняя разработка).

Практически всегда эти два инструмента взаимодействуют друг с другом при описании жизненного цикла документа. Рассмотрим их.

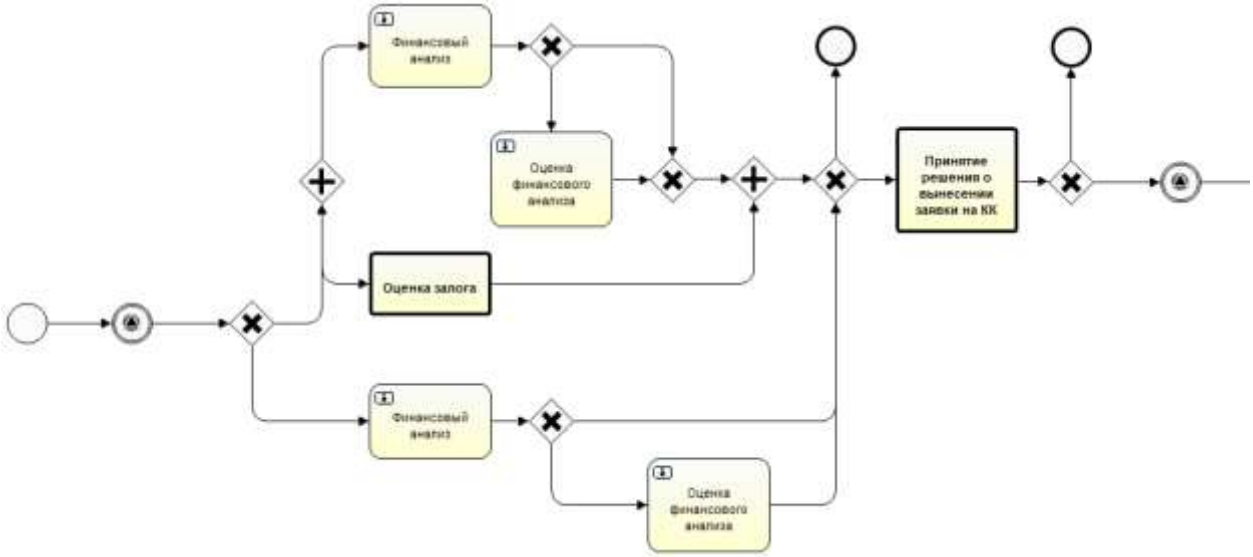
Процессы Activiti

Процессы Activiti – это один из стандартных способов описания процессов в Alfresco. Описание процесса представляет собой xml файл. Вместе с описанием процесса практически всегда есть подчиненная модель данных Alfresco, в которой описаны свойства задач, локализация модели и формы для задач. Не смотря на то что процесс описан в виде простого xml файла, редактировать его в текстовом редакторе неудобно из-за указания координат для графического отображения и из-за текстовых ссылок из одних частей процесса в другие. Для создания/редактирования процесса используется приложение Activiti Designer, представляющее собой плагин для IDE Eclipse.

Ниже некоторые моменты, которые нужно знать об Activiti:

- workflow и process – синонимы;
- definition и instance – разные вещи. • definition – описание процесса (тип), instance – экземпляр запущенного процесса (объект);
- Процесс состоит из задач и переходов;
- У процессов есть свои переменные;
- У задач тоже есть свои переменные (описаны в модели);
- В процессы можно передавать параметры, но это просто начальные значения переменных;
- К задачам можно привязывать формы;
- К процессам также привязаны формы.

Диаграмма процесса, открытого в Activiti Designer, выглядит примерно так:



где:

- Прямоугольники с закругленными углами – задачи пользователям. Процесс пойдет дальше, когда пользователь выполнит задачу.
- Прямоугольники с жирными гранями – вызов подпроцессов;
- Стрелки – переходы;
- Ромб с + внутри – параллельный шлюз. Процесс синхронизирует все входы и идет параллельно по всем выходам;
- Ромб с x внутри – шлюз с выбором выхода. Процесс идет на тот выход, в котором выполняется условие;
- Круг с треугольником внутри – процесс останавливается и ожидает сигнала извне.

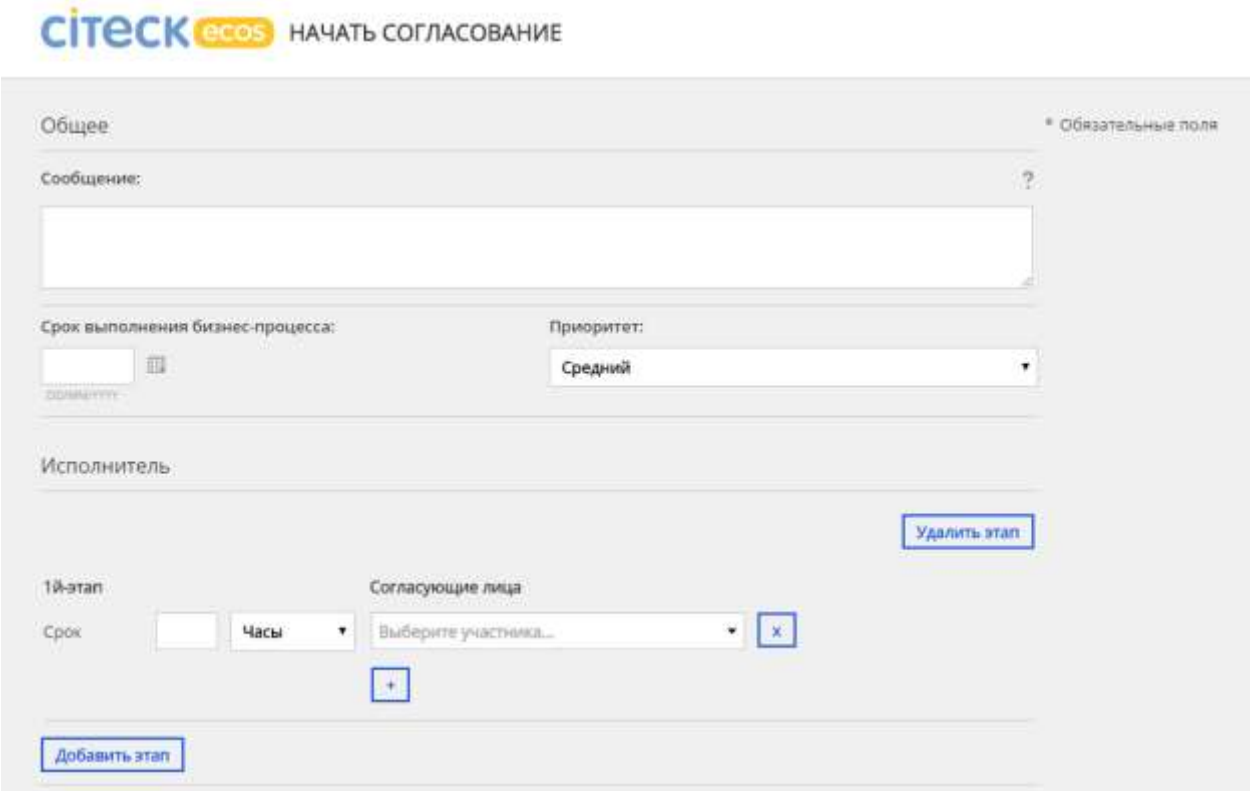
В Citeck EcoS разработан стандартный набор атомарных процессов, которые выполняют одну определенную функцию и удобно между собой комбинируются.

- Согласование (confirm);
- Доп. согласование (additional-confirm);
- Утверждение у контрагента (contractor-approval);
- Подписание (sign);
- Исполнение (perform);
- Регистрация (registration);
- Вынесение резолюции (resolve);
- Исправление (correction);
- Ознакомление (familiar);

Инструменты для работы с процессами activiti

Как уже было сказано, для разработки процессов удобно использовать Activiti Designer. Для управления процессами в Alfresco используется Workflow Console: http://localhost:8080/alfresco/faces/jsp/admin/workflow-console.jsp. Чтобы запустить процесс можно воспользоваться страницей http://localhost:8080/share/page/start-specified-workflow?workflowId=activiti$[definition-id]. Для просмотра информации по процессу можно использовать страницу http://localhost:8080/share/page/workflow-details?workflowId=activiti$[instance-id].

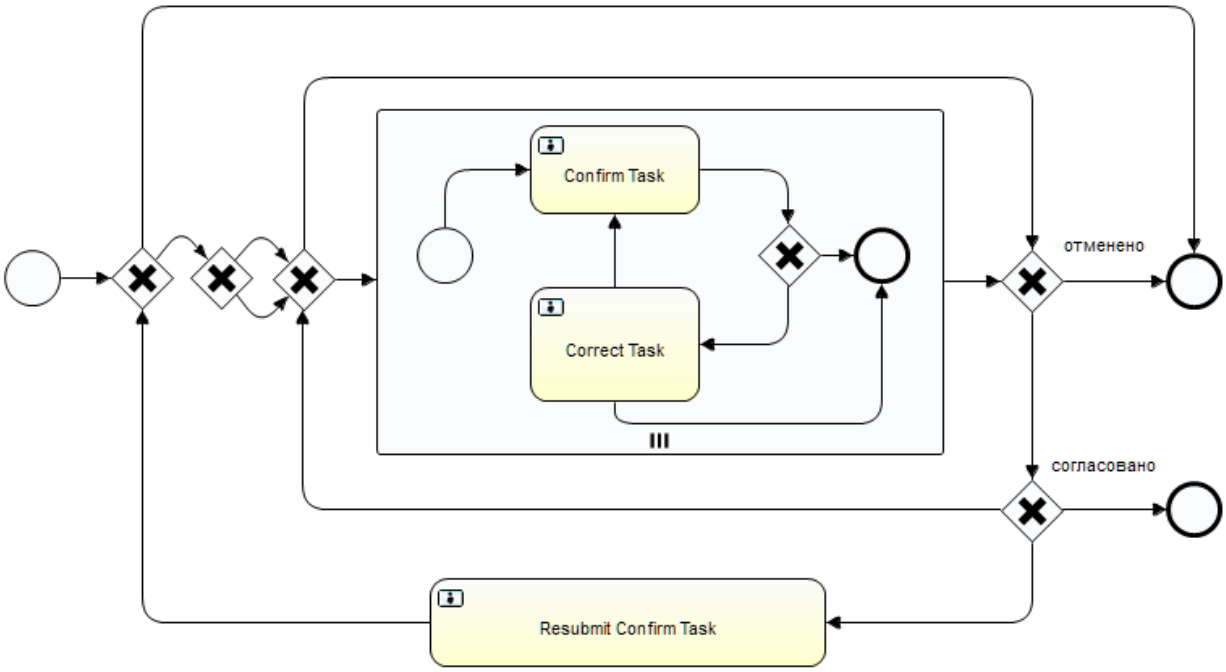
Пример старта процесса согласования: http://localhost:8080/share/page/start-specified-workflow?packageItems=workspace://SpacesStore/...&workflowId=activiti$confirm.



После перехода по ссылке появляется форма процесса согласования. Заполнив обязательные поля и нажав «Начать процесс», мы его запустим.

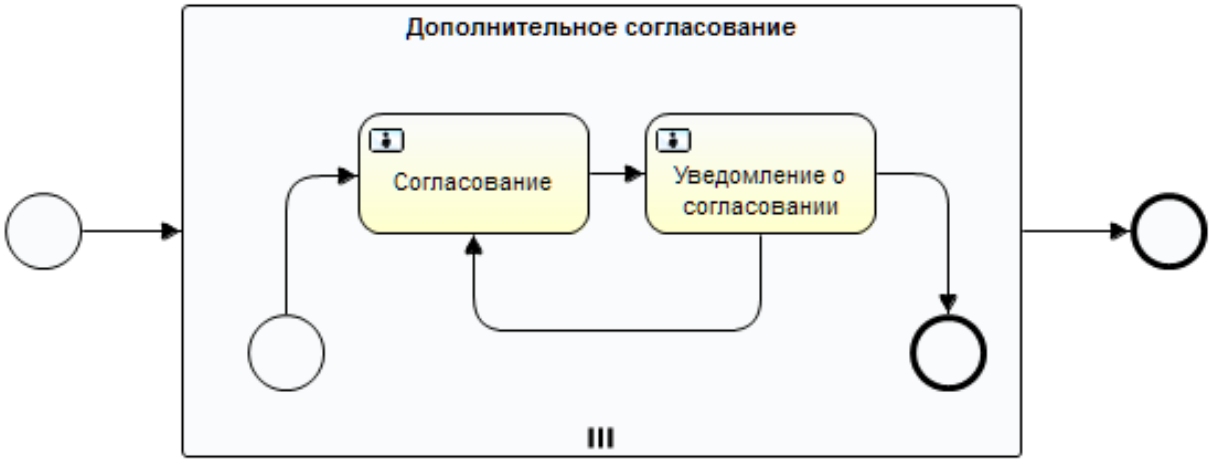
Процессы согласования

Процесс согласования самый распространенный и самый сложный из стандартных процессов Citeck EcoS. Ниже его диаграмма:



На стартовой форме выбираются согласующие. Согласование может быть разбито на несколько этапов. Внутри этапа – определенный список согласующих. Этапы идут последовательно, а согласование внутри этапа происходит параллельно.

Иногда используется процесс доп. согласования. Запускается согласующим прямо из задачи согласования. Имеет следующую диаграмму:



Идентификатор процесса - activiti$additional-confirm. Задача согласования назначается дополнительному согласующему, а Уведомление о согласовании – основному согласующему.

Однозадачные процессы

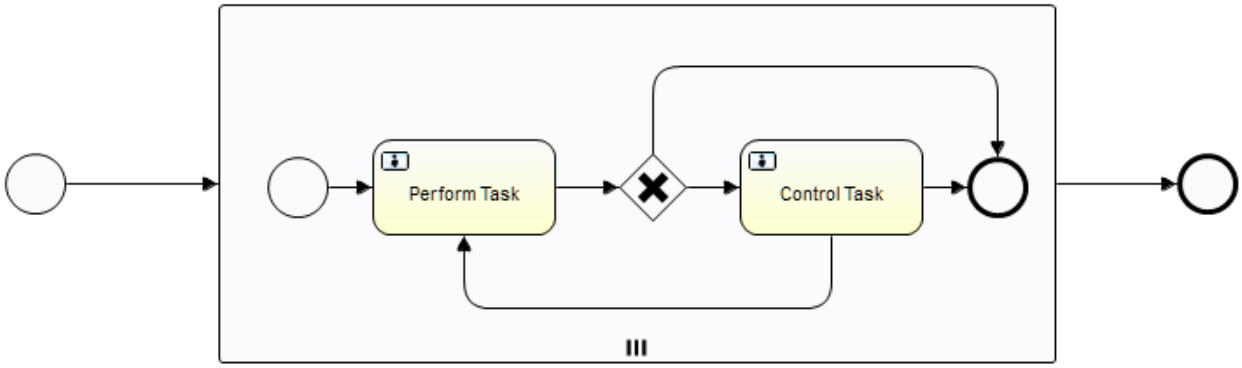
Многие процессы просты и состоят из одной задачи. Так сделано для удобства комбинирования. Ниже процесс подписания:

Идентификатор – activity$sign. Несколько вариантов результата: Подписано, Отклонено, Вернуть на согласование, Вернуть на доработку.

Ниже еще примеры стандартных процессов из одной задачи. Их удобно использовать из жизненного цикла.

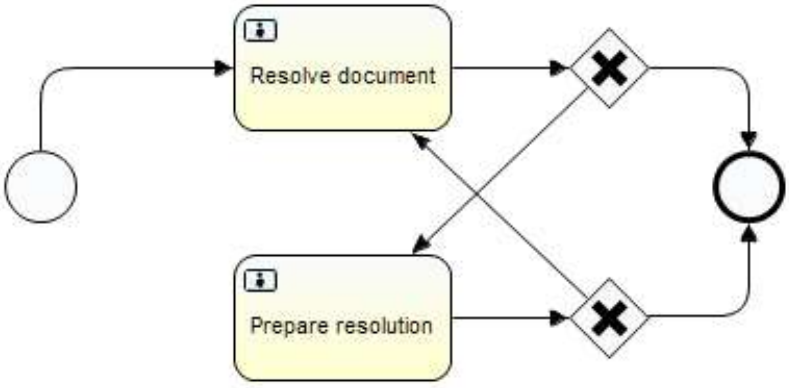
- Исправление – correction;
- Подписание – sign;
- Сканирование – scan;
- Печать – print;
- Регистрация – registration;
- Нормоконтроль – normative-control;
- Пролонгация – prolongation;
- Подтверждение – simple-affirm;
- Оплата – simple-payment;
- Перенос в архив – move-to-archive.

Процесс исполнения



Идентификатор - activiti$perform. 2 задачи: Проверить и Исправить.

Процесс вынесения резолюции



Идентификатор - activiti$resolve. 2 задачи: Подготовить резолюцию, Вынести резолюцию.

Жизненный цикл документа

Функциональность жизненного цикла документа позволяет осуществлять переход документа из состояния в состояние при выполнении определенных условий и привязывать к переходом определенные действия над документом.

Таблица переходов

Переходы документа из состояния в состояние определяются с помощью таблицы переходов. Сама таблица переходов представляет собой csv файл. В дальнейшем планируется переход на формат XML. В системе Citeck EcoS к типу документа может быть привязана одна таблица переходов. Таблица состоит из следующих колонок:

fromState (текст) – обязательная для заполнения колонка. В ней указывается, в каком состоянии должен находиться документ (lc:state), чтобы переход мог быть к нему применен. Значение “*” означает, что переход возможен из любого состояния документа.

event (JSON)– обязательная для заполнения колонка. Содержит объект JSON с основной информацией о переходе. В объекте существуют следующие поля:

- eventType - должен присутствовать обязательно. Определяет тип события, при котором возможен данный переход. Содержит одно из нескольких предопределенных значений:
    - userTransition – переход, осуществляемый при нажатии пользователем на определенное действие в карточке документа;
    - automaticTransition – переход, который осуществляется автоматически. Если документ перешел в состояние, в котором есть переход данного типа, то он старается выполниться сразу же (если transitionCondition позволяет);
    - onEndProcess – переход, который осуществляется при завершении бизнес-процесса определенного типа;
    - onStartProcess - переход, который осуществляется при старте бизнес-процесса определенного типа. Если переход возможен. то в карточке документа (кейса) присутствует действие “Начать бизнесс-процесс”;
    - timerTransition - переход, который осуществляется при прошествии определенного времени;
    - onSignal - Переход, который осуществляется по сигналу из бизнесс-процесса.
- actionName – определяет имя действия, которое будет показано в карточке пользователю для осуществления перехода. Применяется для automaticTransition и onStartProcess.
- workflowId - определяет идентификатор типа процесса. Применяется для onStartProcess и onEndProcess.
- dateTimeExpression - содержит серверный JS, вычисляющий java.util.Date. Используется в timerTransition для вычисления момента осуществления перехода.
- signalId - содержит идентификатор сигнала при котором осуществлять переход. Используется в onSignal.

toState (текст) – Конечное состояние документа при осуществлении перехода.

transitionCondition (JS)– Содержит логическое выражение на javascript. Доступны все возможности серверного JS Alfresco, а также объекты document (текущий документ), person (текущий пользователь) и, если переход с окончанием или стартом процесса, то объект process, содержащий переменные процесса. Также, в JS можно добавлять свои функции, описав их в lifecycle.lib.js. Если JS возвращает true, то данный переход можно осуществлять, в противном случае – нельзя.

action (JS)– JS с возможностями как у transitionCondition, но смысл его, не в возвращаемом значении, а в выполнении определенных действий, сопутствующих переходу.

Как выбирается переход, если возможны несколько вариантов.

Такая ситуация возможна для всех переходов, кроме userTransition. Если из определенного состояния, для определенного типа события, существует несколько переходов, то в первую очередь проверяются переходы с непустым условием (transitionCondition). Выбирается первый, для которого условие выполняется. Если условие не выполняется ни для какого, то выбирается первый переход, в котором transitionCondition пустой. Если такого перехода нет, то никакой переход не осуществляется и документ остается в текущем состоянии.

Загрузка таблицы переходов для типа документа

Для загрузки таблицы переходов сделан вебскрипт. Сама таблица должна быть в Формате csv.

1. CSV файл с таблицей загружается в альфреско;
2. Вызывается вебскрипт /api/lifecycle/create-transition-table?nodeRef={nodeRef}&type={type}

где:
nodeRef – nodeRef csv с таблицей;
type – тип документа, к которому привязать таблицу (например, idocs:inbox).

В будущем будет сделана загрузка и автодеплой через бутстреп и деплой при помещении в определенную папку в рантайме.

Пример таблицы переходов

В качестве примера, создадим таблицу для жизненного цикла, изображенного на картинке:

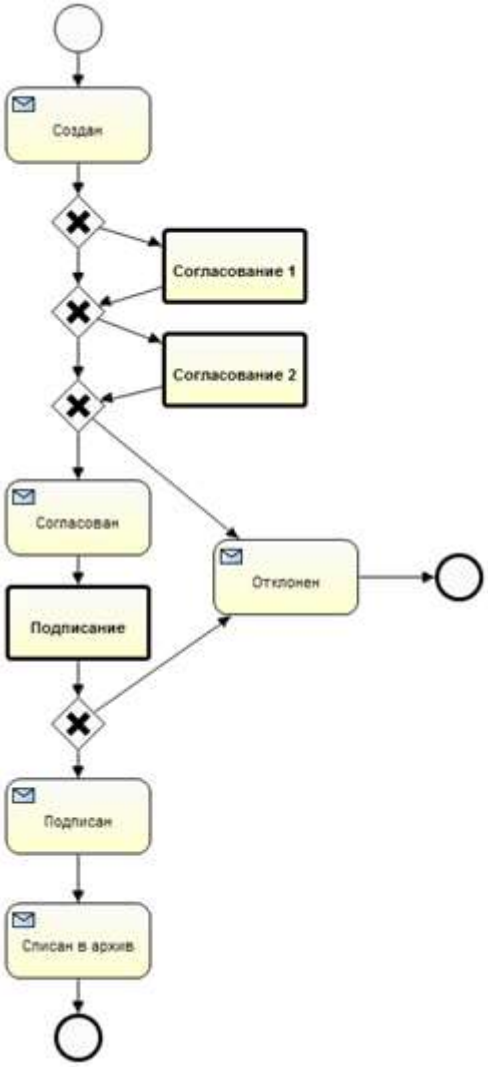


Таблица будет выглядеть следующим образом:

fromState	event	toState	transitionCondition	action
start	{"eventType":"userTransition","dateTimeExpression":"document.properties['cm:modified']","actionName":"Отправить на согласование"}	confirmx1	document.properties.creator == person.properties.userName	
confirmx1	{"eventType":"automaticTransition"}	confirmx2		
confirmx1	{"eventType":"automaticTransition"}	confirm1	(orgstruct.getTypedGroupsForUser(person.properties['cm:userName'], "role", "department_manager").length > 0) || (orgstruct.getTypedGroupsForUser(person.properties['cm:userName'], "role", "branch_manager").length > 0)	startWorkflow ("activiti$confirm", {"wfcf_precedence": person.nodeRef.toString()});
confirm1	{"eventType":"onEndProcess","workflowId":"activiti$confirm"}	declined	process.wfcf_confirmed == false	

confirm1	{"eventType":"onEndProcess","workflowId":"activiti$confirm"}	confirmx2		
confirmx2	{"eventType":"automaticTransition"}	confirmed		
confirmx2	{"eventType":"automaticTransition"}	confirm2	orgstruct.getTypedGroupsForUser(person.properties['cm:userName'], "role", "division_manager").length > 0	startWorkflow ("activiti$confirm", {"wfcf_precedence": person.nodeRef.toString()});
confirm2	{"eventType":"onEndProcess","workflowId":"activiti$confirm"}	declined	process.wfcf_confirmed == false	
confirm2	{"eventType":"onEndProcess","workflowId":"activiti$confirm"}	confirmed		
confirmed	{"eventType":"userTransition","actionName":"Отправить на подписание"}	on-signing	document.properties.creator == person.properties.userName	startWorkflow ("activiti$sign", {"wfsgn_signer": person.nodeRef.toString()});
on-signing	{"eventType":"onEndProcess","workflowId":"activiti$sign"}	declined	process.outcome == 'Declined'	
on-signing	{"eventType":"onEndProcess","workflowId":"activiti$sign"}	signed		
signed	{"eventType":"userTransition","actionName":"Списать в архив"}	archived	orgstruct.getTypedGroupsForUser(person.properties['cm:userName'], "role", "archive_manager").length > 0	