Contents

1. Manual:Extension/Workflows 2
2. Manual:Extension/Workflows/Activity 9
3. Manual:Extension/Workflows/Activity/ApprovePage 9
4. Manual:Extension/Workflows/Activity/CustomForm 10
5. Manual:Extension/Workflows/Activity/EditPage 13
6. Manual:Extension/Workflows/Activity/EditRequest 16
7. Manual:Extension/Workflows/Activity/GroupVote 17
8. Manual:Extension/Workflows/Activity/SendMail 19
9. Manual:Extension/Workflows/Activity/SetTemplateParams 23
10. Manual:Extension/Workflows/Activity/UserVote 26
11. Manual:Extension/Workflows/Triggers 30
12. Manual:Extension/Workflows/Tutorial 33

Workflows

Contents

1 Introduction 3
2 Notifications and reports 3
3 Workflow activities 3
3.1 Single user approval 3
3.2 Expert document control 4
3.3 Group feedback 6
3.4 Feedback 7
4 Overview page 7
4.1 Tasks overview 8
5 How to add a custom workflow 8

Introduction

In BlueSpice 4.1, workflows are based on BPMN 2.0. Four different types of page-based workflows are already integrated. Their purpose is a page review to obtain feedback via a user vote or to trigger a page approval. In the following, these workflows are therefore called review workflows.

Types of reviews

Workflow type	Participants	Description
Single user approval	1 user	A single user is asked to vote about a page. If a user submits a positive vote, the page is automatically approved.
Expert document control	3 users	After a page has been edited by a specific user, the page is reviewed by an expert and then approved by a user who is responsible for approvals.
Group feedback	1 group	A group (which needs to exist in the group manager) is requested to leave a comment on a page.
Single user feedback	1 user	A user is asked to send a comment regarding a page.

Notifications and reports

- **When a task gets assigned:** When a user needs to complete a workflow task, both a notification and a task are created.
- **At the end of a worfklow:** After a workflow has ended, the initiator receives a notification and an email is sent to a report recipient (as defined at the beginning of the workflow).
- **When a workflow gets aborted**:
  - A notifications are sent to users with a pending workflow task
  - A notifications are sent to the originator if a workflow got aborted
  - Participants of past or future steps don't receive a notification

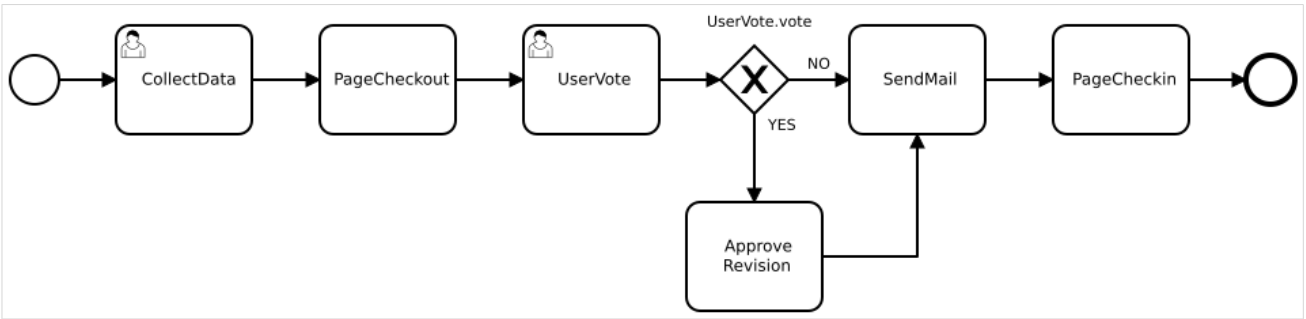
Workflow activities

All approval workflows start with a form where the necessary workflow data is entered by the workflow initiator. Each workflow results in one or more workflow activities.

Single user approval

Purpose: Approval of a draft page by a user with approval rights. This workflow only makes sense if the approval function (FlaggedRevs) is activated on a page.

Workflow instances: Only one approval workflow can run per page.



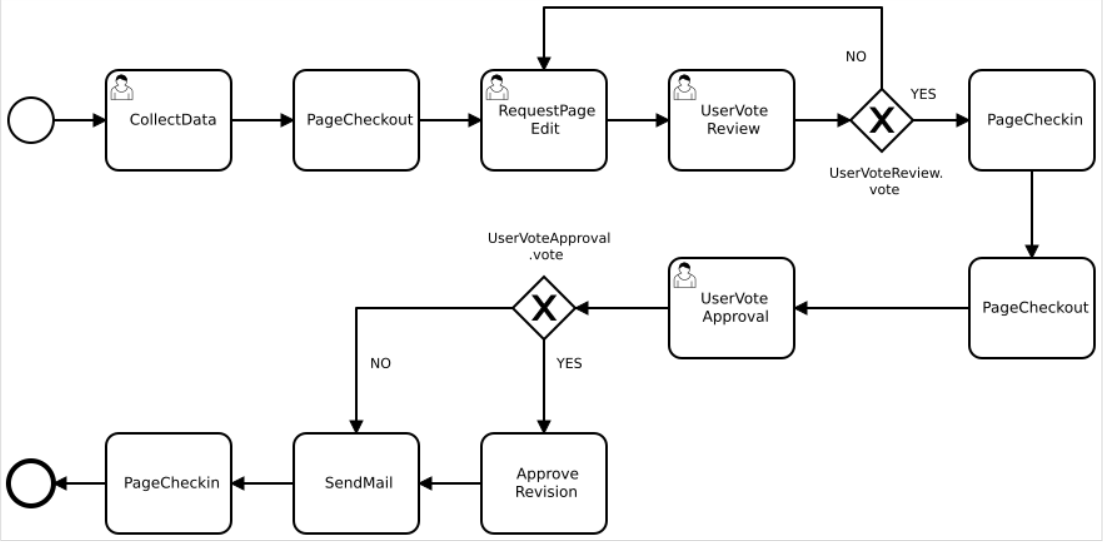
BPMN diagram of a "Single user approval" workflow

Activity	Description
CollectData	In the first workflow step, the workflow initiator enters the settings: • *User:* ser who is assigned to the task. • *Instructions:* A comment or instructions for the user to understand the task. • *Send report to:* An email report with the results will be sent to this email address when the review is finished. If a username is specified here, an email address must be stored in the user administration so that the report can be sent.
PageCheckout	The page is locked for editing by other wiki users. Only the user with the workflow task can edit while this workflow is running.
UserVote	The assigned user carries out a vote and either accepts the page or rejects it. Alternatively, the task can be delegated. In the event of a rejection, the workflow skips the next step (ApprovePage).
ApprovePage	**Only if** the user has submitted a positive vote (accept), the page is set to an approved state.
SendMail	An email report is sent to the report recipient who was specified in the first step.
PageCheckin	The page is unlocked.

Expert document control

Purpose: Approval of a draft page according to the "4-eyes principle".

Workflow instances: A page can only have one approval worfklow at a time.



BPMN diagram of the "Expert document control" workflow

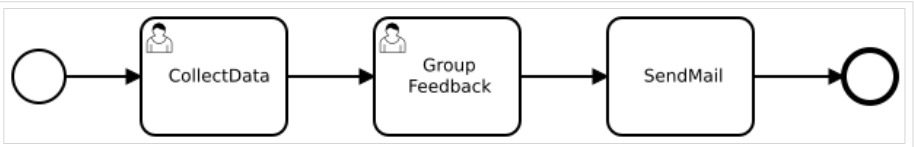
Activity	Description
CollectData	In the first workflow step, the workflow initiator enters the settings: *User:* User who is assigned to a task. Three different users have to be specified: **Editor**, **Reviewer**, **Approver** *Instructions:* A comment or instructions for the users to understand their tasks. *Send report to:* An email report with the results will be sent to this email address when the review is finished. If a username is specified here, an email address must be stored in the user administration so that the report can be sent.
PageCheckout	The page is locked for users who do not participate in the workflow. Only the **Editor** (first workflow participant) and **Reviewer** (second participant) of the workflow can edit the page during checkout.
EditPage	The **Editor** user can edit the page and completes the task without comment.
UserVote	After the **Editor** step has been completed,the **Reviewer** user can edit the page and submit a vote. As an alternative, the **Reviewer** can delegate the task. If the vote is positive (Approve), the workflow continues. If the **Reviewer** rejects, the workflow goes back to the **Editor**.
PageCheckin	After the **Reviewer** submits a positive vote (accept), the page is checked in and the workflow continues.

Activity	Description
PageCheckout	In this step, the page checkout locks the page for editing completely. The **Approver** user will not be able to change the page, but needs to approve it.
ApprovePage	The Approver can either complete or delegate the task. After the **Approver** (or the delegate) finishes the assigned task, the page is set from "draft" to "approved" status if the page was in draft status (only if the approver submits a positive vote). If not, this step is skipped.
SendMail	If an email or user was specified in the workflow settings, the report is now getting sent to that user.
PageCheckin	The page gets unlocked for editing.

Group feedback

Purpose: Obtaining feedback from the members of a user group. The group must exist in the group manager.

Workflow instances: Several feedback workflows can run independently of one another on one page at the same time.



BPMN diagram of the "Group feedback" workflow

Activity	Description
CollectData	In the first workflow step, the workflow initiator enters the settings: • *Group:* User group who is assigned to the task. • *Instructions:* A comment or instructions for the users to understand their task. • S*end report to:* An email report with the results will be sent to this email address when the review is finished. If a username is specified here, an email address must be stored in the user administration so that the report can be sent.
GroupFeedback	All users in the assigned group provide feedback via a comment field. This is a parallel workflow, which means that the order of the feedback does not matter.
SendMail	If an email or user was specified in the workflow settings, the report is now getting sent to that user.

Feedback

Purpose: Obtaining feedback from a single user on a page.

Workflow instances: Multiple feedback worfklows can run independently of one another on one page at the same time.

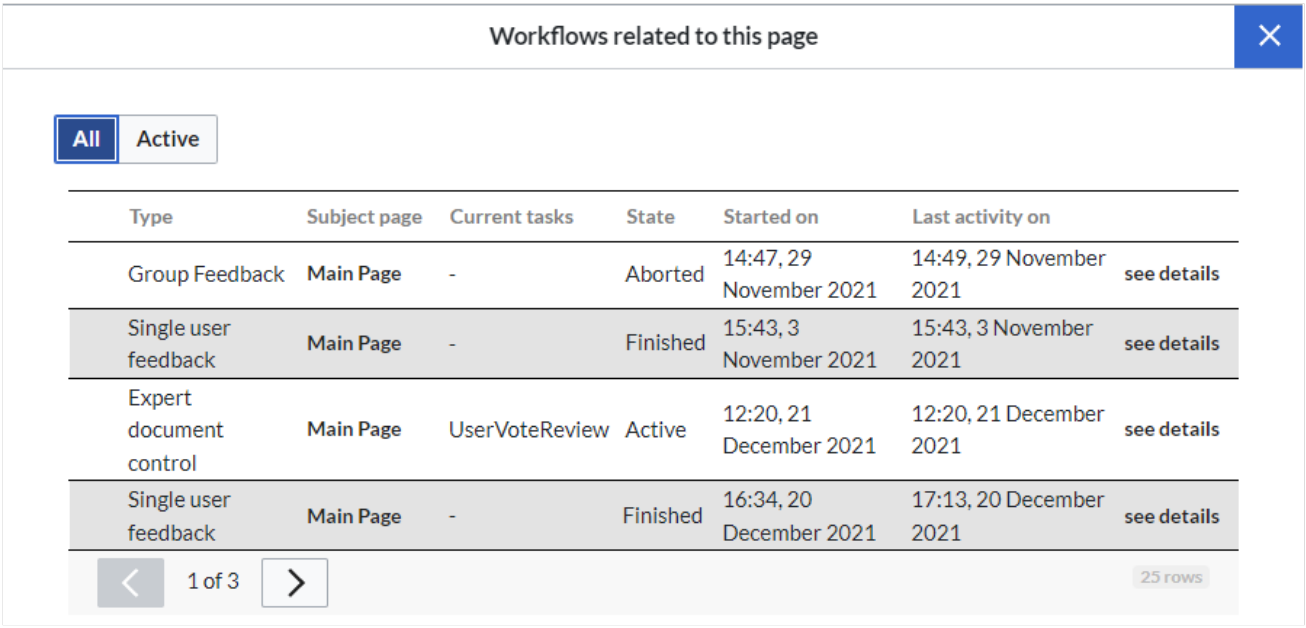


BPMN diagram of the "Single user feedback" workflow

Activity	Description
CollectData	In the first workflow step, the workflow initiator enters the settings: • *User:* User who is assigned to the task. • *Instructions:* A comment or instructions for the users to understand their task. • Send report to: An email report with the results will be sent to this email address when the review is finished. If a username is specified here, an email address must be stored in the user administration so that the report can be sent.
UserFeedback	The assigned user sends a comment.
SendMail	If an email or user was specified in the workflow settings, the report is now getting sent to that user.

Overview page

All workflows in the wiki are listed on the page *Special:Workflows overview*. A view for all *active* workflows and a view for *all* workflows can be selected.



Workflows overview page

Tasks overview

Users are informed about new and pending tasks in their notifications. They can view assigned workflows on their *My tasks* page.

How to add a custom workflow

Users can upload an xml-file of a BPMN diagram with custom activities to the wiki. Currently, the following predefined activities exist:

Extension: Workflows

- CustomForm
- UserVote
- GroupVote
- UserFeedback
- GroupFeedback
- SendMail
- EditRequest

Extension: PageCheckout

- PageCheckOut
- PageCheckIn

Extension: BlueSpiceFlaggedRevsConnector

- ApprovePage

Example of a customized workflow (coming soon)

Technical Reference: Workflows

Manual:Extension/Workflows/Activity

Workflow activities that currently can be included in a workflow:

- ApprovePage
- CustomForm
- EditPage
- EditRequest
- GroupVote
- SendMail
- SetTemplateParams
- UserVote

Manual:Extension/Workflows/Activity/ApprovePage

Description

The ApprovePage activity will set the status of a page from *draft* to *approved* status. Flagging will be done with a special user-context to avoid permission errors. This activity can ony be used with pages that have the approval feature (FlaggedRevs) enabled.

Short profile	
Name	ApprovePage
Async	Yes
BPMN type	`bpmn:Task`
BPMN Extension Element "wf: type"	`approve_revision`

Properties

Name of property	Description	Type
`comment`	The comment to be saved with the approval	string
`revision`	The revision ID to approve. If not provided explicitly, the page revision of the workflow context will be used, which may be outdated.	int

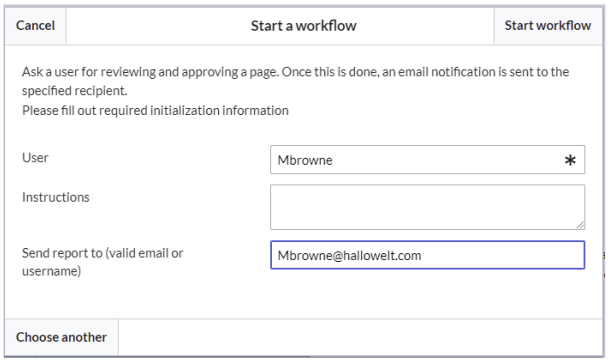
Manual:Extension/Workflows/Activity/CustomForm

Contents

1 Description 11
2 Extension elements 11
2.1 Example 12
3 Properties 12
3.1 Example 12

Description

The *CustomForm* activity is a generic activity, which is based on a custom form. It collects data for the corresponding workflow, which shall be executed. Therefore, the input fields depend on the type of workflow, which is chosen by the user. In general, the data is always the same and differs only in relation to the particular workflow and the number of users required. The *CustomForm* activity mostly include blank values, which can also be inserted by default. This is related to the metadata maintenance associated with the use of wikitext.



CustomForm

Short profile	
Name	CustomForm
Async	No
Input/form	*Input depends on the workflow type, but in general:* • User picker / Group picker • Textfield for comments • Textfield for mail / username • ...
BPMN Element	`bpmn:userTask`
BPMN Extension Element "wf: type"	`custom_form`

Extension elements

Name of extension element	Description	Type
formModule		element
formModule/module	ResourceLoader module to be loaded	string
formModule/class	Form definition class to be shown	string

Example

```
<bpmn:extensionElements>
    <wf:type>custom_form</wf:type>
    <wf:formModule>
                <wf:module>my.custom.form</wf:module>
                <wf:class>MyCustomForm</wf:class>
        </wf:formModule>
</bpmn:extensionElements>
```

This would load the ResourceLoader module called `my.custom.form` and expect it to have a JavaScript class `MyCustomForm` be implemented. Such a RL module may be provided by an extension, or a custom gadget.

Properties

Name of property	Source	Description	Type
`due_date`	UIActivity	Due date for task completion	date/timestamp

Each field provided by the form specified in `formModule/class` needs to be listed as a property with a default value.

Example

If the form specified in `<wf:class>` looks like this

```
MyCustomForm = function( cfg, activity ) {
        MyCustomForm.parent.call( this, cfg, activity );
};

OO.inheritClass( MyCustomForm, workflows.object.form.Form );

MyCustomForm.prototype.getDefinitionItems = function() {
        return [
                {
                        name: 'username',
                        label: "Username",
                        type: 'user_picker',
                        required: true
                },
                {
                        name: 'comment',
                        label: 'Comment'
                        type: 'wikitext',
                }
        ];
};
```

the properties can look like this

```
<bpmn:property name="username" />
<bpmn:property name="comment" default="Some default comment text" />
```

Given the activities `id` specified in the `<bpmn:userTask>` element was `"CollectData"`, connected activities would be able to access the output data in their respective properties like this:

```
<bpmn:task id="SomeActivity">
    ...
    <bpmn:property name="user" default="{{{CollectData.username}}}"></bpmn:property>
    <bpmn:property name="intro" default="{{{CollectData.comment}}}"></bpmn:property>
    ...
```

Manual:Extension/Workflows/Activity/EditPage

Contents

1 Description 14
2 Properties 14
2.1 Input properties 14
2.2 Output properties 15
3 Element 15
4 Example 15

Description

The *EditWikiPage* activity appends content to an existing wiki page.

Short profile	
Name	EditWikiPage
Async	Yes
BPMN type	`bpmn:Task`
BPMN Extension Element "wf: type"	`edit_page`

Properties

Input properties

Each input property can provide a default value if no other value is set by the workflow.

Name of property	Description	Possible values	Type
`title`	Name of the wiki page that will be edited by this activity.		string
`user`	Name of user that will be shown as making the edit.	Default value: *MediaWiki default*	string
`content`	Content that will be added to the wiki page.		string
`mode`	How the content is added to the wiki page.	• *append:* Content is added at the end of the page. • *replace:* Content replaces all existing page content. • *prepend:* Content is added at the beginning of the page.	string
`minor`	The edit will be marked as *major* or *minor* revision.	• *1* (=minor revision) • *0* (=major revision)	int

Output properties

Name of property	Description	Type
`revisionId`	The revision ID of the edited page.	string
`timestamp`	The revision time of the edited page.	timestamp

Element

```
<bpmn:task id="Activity_18egh9o" name="Edit page">
    <bpmn:extensionElements>
                <wf:type>edit_page</wf:type>
        </bpmn:extensionElements>
        <bpmn:property name="title" default="Some page" validation="required"/>
        <bpmn:property name="user" default="Mediawiki default" validation="existing-
user"/>
        <bpmn:property name="content" default="This is my test at editing"/>
        <bpmn:property name="mode" default="append"/> <!-- ALlowed: append, replace,
prepend -->
        <bpmn:property name="minor" default="1"/>
        <!-- Output properties -->
        <bpmn:property name="revisionId"/>
        <bpmn:property name="timestamp"/>
</bpmn:task>
```

Example

In this example, when the workflow runs on any wikipage, The page Testpage1 is appended with the text `<username of initiator> read the page.`

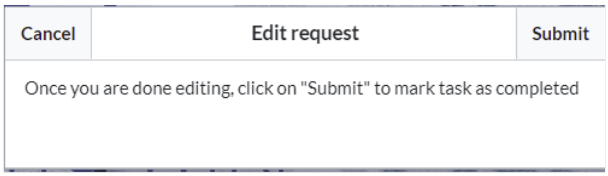
```
<?xml version="1.0" encoding="UTF-8"?>
<bpmn:definitions xmlns:bpmn="http://www.omg.org/spec/BPMN/20100524/MODEL" xmlns:wf="ht
tp://hallowelt.com/schema/bpmn/wf">
    <bpmn:process id="EditWikiPage-Workflow" isExecutable="false">
        <bpmn:extensionElements>
            <wf:context>
                <wf:contextItem name="pageId"/>
            </wf:context>
        </bpmn:extensionElements>
        <bpmn:startEvent id="TheStart">
            <bpmn:outgoing>FromTheStartToEditWikiPage</bpmn:outgoing>
        </bpmn:startEvent>
        <bpmn:sequenceFlow id="FromTheStartToEditWikiPage" sourceRef="TheStart" targetR
ef="EditWikiPage" />
       <bpmn:task id="EditWikiPage" name="Edit page">
    <bpmn:extensionElements>
                <wf:type>edit_page</wf:type>
        </bpmn:extensionElements>
        <bpmn:property name="title" default="Testpage1" validation="required"/>
        <bpmn:property name="user" default="{{{initiator}}}" validation="existing-user"
/>
        <bpmn:property name="content" default="{{{initiator}}} read the page"/>
        <bpmn:property name="mode" default="append"/> <!-- ALlowed: append, replace,
prepend -->
        <bpmn:property name="minor" default="0"/>
        <!-- Output properties -->
        <bpmn:property name="revisionId"/>
```

```
        <bpmn:property name="timestamp"/>
          <bpmn:incoming>FromTheStartToEditWikiPage</bpmn:incoming>
          <bpmn:outgoing>FromEditWikiPageToTheEnd</bpmn:outgoing>
</bpmn:task>
        <bpmn:sequenceFlow id="FromEditWikiPageToTheEnd" sourceRef="EditWikiPage" targe
tRef="TheEnd" />
        <bpmn:endEvent id="TheEnd">
            <bpmn:incoming>FromEditWikiPageToTheEnd</bpmn:incoming>
        </bpmn:endEvent>
    </bpmn:process>
</bpmn:definitions>
```

Manual:Extension/Workflows/Activity/EditRequest

Description

The *EditRequest* activity asks a specified user to edit a page where a workflow is currently running. This activity consists of a form that simply provides information about editing the corresponding page and then submitting the change.



EditRequest dialog

Short profile	
Name	EditRequest
Async	No
BPMN type	`bpmn:userTask`
BPMN Extension Element "wf: type"	`edit_request`

Properties

Name of property	Source	Description	Type
`due_date`	UIActivity	Due date for task completion	date /timestamp
		Name of the user that should edit a page. Can be	

Name of property	Source	Description	Type
`assigned_user`	UIActivity	plain username (e.g. "WikiSysop") or user page ("User:WikiSysop"); Support for User-ID is not required	string
`instructions`	InstructedActivity	Text that is shown to the user, so he knows what to do; visible in the UI / form but is optional	string

Manual:Extension/Workflows/Activity/GroupVote

Description

The activity *GroupVot*e is responsible for collecting necessary data about the voting of a group on a special topic. Unlike the UserVote, it is not possbile to delegate the task. A user that is part of a group can accept or decline a vote and also leave a comment that justifies their decision. The voting result is determined by whatever threshold has been reached first and ends the activity.

Short profile	
Name	GroupVote
Async	No
BPMN type	`bpmn:userTask`
BPMN Extension Element "wf: type"	`group_vote`

Properties

Name of property	Description	Type
`due_date`	Due date for task completion	date /timestamp
`assigned_group`	Name of the user group that should vote; can be plain grouname as used in the DB (e.g. "sysop")	string

Name of property	Description	Type
`instructions`	Text that is shown to the group of user, so they know what to vote about	string

`users_voted`	Not to be set in the workflow definition. Used to store data during the activities life cycle. E.g.
	`[ { "userName": "UserA", "vote": "yes", "comment": "Good" }, { "userName": "UserB", "vote": "no", "comment": "Not good" } ]`
	Can be accessed by follow up activitites by e.g.
	`{{<GroupVoteActivityID>.0.userName}}`
		string

`threshold_yes_unit`	can be `user` or `percent` - absolute number of users required to a approve a page	user, percent
`threshold_yes_value`	number of users or percentage of users required to approve a page	
`threshold_no_unit`	can be `user` or `percent` - absolute number of users required to a reject a page	user, percent
`threshold_no_value`	number of users or percentage of users required to reject a page	

Full example

Once one of the thresholds is reached, the activity will be completed and a `<bpmn:exclusiveGateway>` (referenced by `<bpmn:outgoing>`) will be called. It will choose its outgoing `<bpmn:sequenceFlow>` by mapping its `name` to the "type" of the threshold that has been reached.

In this example, after the first user either voted yes or no, the workflow will simply reach the end point:

```
<?xml version="1.0" encoding="UTF-8"?>
<bpmn:definitions xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:wf="http:
//hallowelt.com/schema/bpmn/wf" xmlns:bpmn="http://www.omg.org/spec/BPMN/20100524
/MODEL" xmlns:bpmndi="http://www.omg.org/spec/BPMN/20100524/DI" xmlns:dc="http://www.
omg.org/spec/DD/20100524/DC" xmlns:di="http://www.omg.org/spec/DD/20100524/DI" id="Defi
nitions_1vrglfw" targetNamespace="http://bpmn.io/schema/bpmn" exporter="bpmn-js
(https://demo.bpmn.io)" exporterVersion="8.7.1">
        <!-- Process part -->
        <bpmn:process id="Group_Vote" isExecutable="false">
                <bpmn:extensionElements>
                        <wf:context>
                                <wf:contextItem name="pageId"/>
                            <wf:contextItem name="revision"/>
                        </wf:context>
```

```
                </bpmn:extensionElements>
                <!-- StartEvent -->
                <bpmn:startEvent id="TheStart">
                        <bpmn:outgoing>FromTheStartToGroupVote</bpmn:outgoing>
                </bpmn:startEvent>
                <bpmn:sequenceFlow id="FromTheStartToGroupVote" sourceRef="TheStart" ta
rgetRef="GroupVote"/>
                        <!-- Collect group vote -->
                <bpmn:userTask id="GroupVote" name="GroupVote">
                        <bpmn:extensionElements>
                                <wf:type>group_vote</wf:type>
                        </bpmn:extensionElements>
                        <bpmn:property name="assigned_group">QM-reviewer</bpmn:
property>
                        <bpmn:property name="instructions">Please vote</bpmn:property>
                        <bpmn:property name="comment"/>
                        <bpmn:property name="vote">vote</bpmn:property>
                        <bpmn:property name="users_voted"/>
                        <bpmn:property name="due_date">{{#time:YmdHis|now + 7days}}<
/bpmn:property>

                        <bpmn:property name="threshold_yes_unit">user</bpmn:property>
                        <bpmn:property name="threshold_yes_value">1</bpmn:property>
                        <bpmn:property name="threshold_no_unit">user</bpmn:property>
                        <bpmn:property name="threshold_no_value">1</bpmn:property>

                        <bpmn:incoming>FromtheStartToGroupVote</bpmn:incoming>
                        <bpmn:outgoing>FromGroupVoteToGatewayGroupVote</bpmn:outgoing>
                </bpmn:userTask>
                <bpmn:sequenceFlow id="FromGroupVoteToGatewayGroupVote" sourceRef="Grou
pVote" targetRef="GatewayGroupVote"/>
                <!-- Check on voting -->
                <bpmn:exclusiveGateway id="GatewayGroupVote" name="GroupVote.vote">
                        <bpmn:incoming>FromGroupVoteToGatewayGroupVote</bpmn:incoming>
                        <bpmn:outgoing>FromGatewayGroupVoteToTheEndYes</bpmn:outgoing>
                   <bpmn:outgoing>FromGatewayGroupVoteToTheEndNo</bpmn:outgoing>
                </bpmn:exclusiveGateway>
                <bpmn:sequenceFlow id="FromGatewayGroupVoteToTheEndYes" name="yes" sour
ceRef="GatewayGroupVote" targetRef="TheEnd"/>
                <bpmn:sequenceFlow id="FromGatewayGroupVoteToTheEndNo" name="no" source
Ref="GatewayGroupVote" targetRef="TheEnd"/>
                <!-- EndEvent -->
                <bpmn:endEvent id="TheEnd">
                        <bpmn:incoming>FromGatewayGroupVoteToTheEndYes</bpmn:incoming>
            <bpmn:incoming>FromGatewayGroupVoteToTheEndNo</bpmn:incoming>
                </bpmn:endEvent>
        </bpmn:process>
</bpmn:definitions>
```

Manual:Extension/Workflows/Activity/SendMail

Contents

1 Description 21
2 Properties 22
2.1 Input 22
2.2 Output 22

3 Extension elements 23

Description

The *SendMail* activity is a very generic activity. It is meant to send a report via mail after an action has been completed. This could be done after voting, but may also concern other activities in future, for example.

Following a quick overview:

Short profile	
Name	SendMail
Async	No
Input/form	-
Associated to	Approval Workflows, Collect Feedback Workflows
BPMN type	bpmn:Task

Properties

Input

Name of property	Source	Description	Type	Action
assigned_user	UserVote	Name of the user, who casted the vote	string	collect
comment	UserVote	Comment of the user	string	collect
vote	UserVote	Result of the voting (values: YES, NO)	string	collect
reportrecipient	CollectData	Mailadress of the user who was specified to receive an email after the workflow is finished	string	collect
timestamp		Timestamp of the date when mail is sent	timestamp	display
timestamp		Timestamp of the voting	timestamp	display
username		Name of the user, who casted a vote	string	display
groupname		Name of the group, who casted a vote	string	display
vote		Result of the voting	boolean / string	display
comment		Comment of the user concerning a wikipage	string	display

Output

Name of property	Source	Description	Type	Action
subject	SendMail	Subject of the mail incl. corresponding wikipage affected by previous activity	string	display
body	SendMail	Standardized text in a mail, which includes • the name of the user who casted the vote • a comment, which was left by the user • the voting result	string	display

Extension elements

Name of extension element	Description	Type
type		

Manual:Extension/Workflows/Activity/SetTemplateParams

Contents

1 Description 24
2 Profile 24
3 Properties 24
4 Example 25
4.1 Workflow task 25
4.2 Template 25
4.3 Full bpmn example 26

Description

The *SetTemplateParams* activity allows automatic setting a template parameter on a wiki page.

- Templates are accessed by the order of appearance on the page, starting at 0. Non-existing templates are counted as links, so they do not count in this list.
- Parameters inside the template can be specified by their name or the index in case of non-named parameters.
- Index for non-named params starts at 1 (not 0!), as to be consistent with how template params are accessed in the template itself.
- Nested templates are currently not supported.

Profile

Short profile	
Name	SetTemplateParams
Async	Yes
BPMN type	`bpmn:task`
BPMN Extension Element "wf: type"	`set_template_param`

Properties

Name of property	Description	Type
`title`	Name of the page where the template parameter is changed. If it will be changed on the page where the workflow runs, the value is `{{FULLPAGENAME}}`	string
`user`	User that is shown in the revision history	user
`template-index`	Index of the template on the page (starting at 0)	number
`template-param`	Parameter to modify. This can be a string (in the case of a named parameter) or a number (in case of a non-named param; starting from 1, accessor for the parameter)	int|string
`value`	New value of the parameter, set by the workflow	string
`minor`	Marks the revision as a minor edit if set to 1	boolean

Name of property	Description	Type
`revisionId`	Sets a new page revision ID	number
`timestamp`	Sets the page revision timestamp	timestamp
`comment`	Comment that is shown in the revision history	string

Example

Workflow task

This workflow task will set the value of the parameter `{{{status|not set}}}` to `Updated` on a page's first template.

```
<bpmn:task id="EditTemplate" name="Edit template">
            <bpmn:extensionElements>
                        <wf:type>set_template_param</wf:type>
                </bpmn:extensionElements>
                <bpmn:property name="title" default="Testpage" validation="required,
existing-title"/>
                <bpmn:property name="user" default="WikiSysop" validation="existing-
user"/>
                <!-- Index of the template on page (starting from 0) -->
                <bpmn:property name="template-index" default="0"/>
                <!-- Param to modify. Can be a string in case of named params, or a
number, in case of non-named params (starting from 1, accessor for the param) -->
                <bpmn:property name="template-param" default="status"/>
                <!-- New value to set -->
                <bpmn:property name="value" default="Updated"/>
                <bpmn:property name="minor" default="0"/>

                <!-- Output properties -->
                <bpmn:property name="revisionId"/>
                <bpmn:property name="comment" default="Edited template param"/>
                <bpmn:property name="timestamp"/>
      <bpmn:incoming>Flow_1qh6vpl</bpmn:incoming>
      <bpmn:outgoing>Flow_0if1kyp</bpmn:outgoing>

    </bpmn:task>
```

Template

The template content has the parameter `{{{status}}}` At the time of writing this, the workflow produced an error tif the parameter was wrapped in a `<div>` tag. Therefore, in the example below, the parameter is defined in a variable first. If the parameter is not in a `div` tag, using a variable is not necessary.

```
{{#vardefine:status|{{{status|not set}}}}}
<div style="background:#fbfaef;">Page status: {{#var:status}}</div>
```

Full bpmn example

```
<?xml version="1.0" encoding="UTF-8"?>
<bpmn:definitions xmlns:bpmn="http://www.omg.org/spec/BPMN/20100524/MODEL" xmlns:wf="ht
tp://hallowelt.com/schema/bpmn/wf">
    <bpmn:process id="SetTemplateParams-Process" isExecutable="false">
        <bpmn:extensionElements>
            <wf:context>
                <wf:contextItem name="pageId"/>
            </wf:context>
        </bpmn:extensionElements>
        <bpmn:startEvent id="TheStart">
            <bpmn:outgoing>FromTheStartToEditTemplate</bpmn:outgoing>
        </bpmn:startEvent>
        <bpmn:sequenceFlow id="FromTheStartToEditTemplate" sourceRef="TheStart" targetR
ef="EditTemplate" />
        <bpmn:task id="EditTemplate" name="Edit template">
            <bpmn:extensionElements>
                <wf:type>set_template_param</wf:type>
            </bpmn:extensionElements>
            <bpmn:property name="title" default="" validation="required,existing-title"
>{{FULLPAGENAME}}</bpmn:property>
            <bpmn:property name="user" default="WikiSysop" validation="existing-user">W
ikiSysop</bpmn:property>
            <!-- Index of the template on page (starting from 0) -->
            <bpmn:property name="template-index" default="0"/>
            <!-- Param to modify. Can be a string in case of named params, or a
number, in case of non-named params (starting from 1, accessor for the param) -->
            <bpmn:property name="template-param" default="status"/>
            <!-- New value to set -->
            <bpmn:property name="value" default="Super edited param"/>
            <bpmn:property name="minor" default="1"/>
            <!-- Output properties -->
            <bpmn:property name="revisionId"/>
            <bpmn:property name="timestamp"/>
            <bpmn:incoming>FromTheStartToEditTemplate</bpmn:incoming>
            <bpmn:outgoing>FromEditTemplateToTheEnd</bpmn:outgoing>
        </bpmn:task>
        <bpmn:sequenceFlow id="FromEditTemplateToTheEnd" sourceRef="EditTemplate" targe
tRef="TheEnd" />
        <bpmn:endEvent id="TheEnd">
            <bpmn:incoming>FromEditTemplateToTheEnd</bpmn:incoming>
        </bpmn:endEvent>
    </bpmn:process>
</bpmn:definitions>
```

Manual:Extension/Workflows/Activity/UserVote

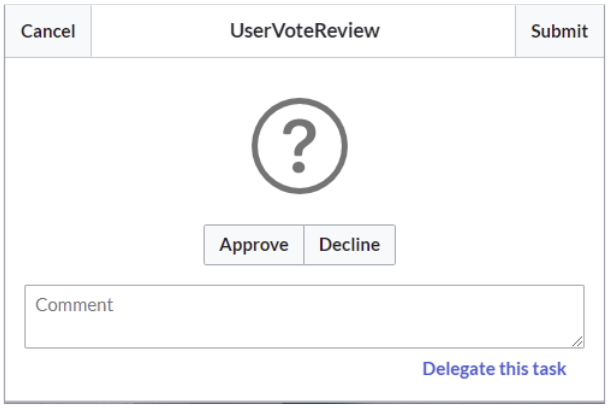
Contents

1 Description 28
2 Extension elements 29
3 Properties 29
3.1 Input 29

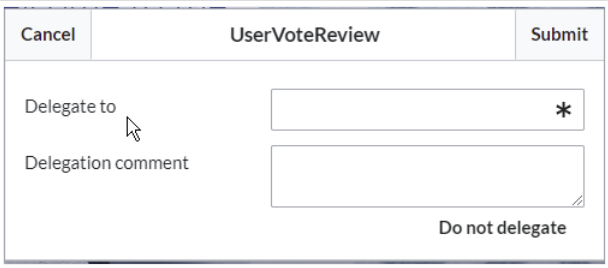
3.2 Output 30

Description

The activity *UserVote* is responsible for collecting necessary data about the voting of a user on a special topic. The user who casts a vote has three options: accept, decline and delegate. If a user chooses "delegate", a dialog with a "UserPicker" must be provided. The chosen user must be notified and allowed to vote. The delegatee can re-delegate, but only to the primary user. A notification must be sent again. All delegation actions of the activity must be logged internally and be passed to the next activity.



Dialog for user voting



Dialog for delegating

Short profile	
Name	UserVote
Async	No

Input/form	• Submit / Cancel button
	• Voting button (approve/decline)
	• Textfield for comments
	• Link for delegating the task
	  ○ Userpicker
	  ○ Textfield for comments

Associated to	• ApprovalWorkflows • FeedbackWorkflows
BPMN type	bpmn:userTask

Extension elements

Name of extension element	Description	Type
form	Name of a *.form page on the wiki	string

Properties

Input

Name of property	Source	Description	Type	Action
assigned_user	CollectData	Name of the user that should vote. Can be plain username (e.g. "WikiSysop") or user page ("User:WikiSysop"); Support for User-ID is not required	string	none
instructions	CollectData	Text that is shown to the user, so he knows what to vote about; visible in the UI / form	string	display
due_date	CollectData	Due date for task completion	date /timestamp	none
delegate_to	UserVote	Name of the delegated user, that should vote instead of the specified user	string	collect & display
delegate_comment	UserVote	Text that is shown to the delegated user, so he knows what to vote about; visible in the UI / form	string	collect & display

Output

Name of property	Source	Description	Type	Action
vote	UserVote	Result of the voting (values: YES, NO)	boolean /string	collect
comment	UserVote	Comment of the user	string	collect
timestamp	UserVote	Timestamp of the vote	timestamp	collect
revisionid	UserVote	Revision ID that was voted on	integer	collect

Manual:Extension/Workflows/Triggers

The special page *MediaWiki:WorkflowTriggers* makes it possible to set up different triggers for existing workflows. You can access this page from *Global actions > Tools > Workflow triggers.*

Contents

1 Types of triggers 31
1.1 Based on a semantic property 31
1.2 On edit 33
1.3 On page creation 33
1.4 Manual 33

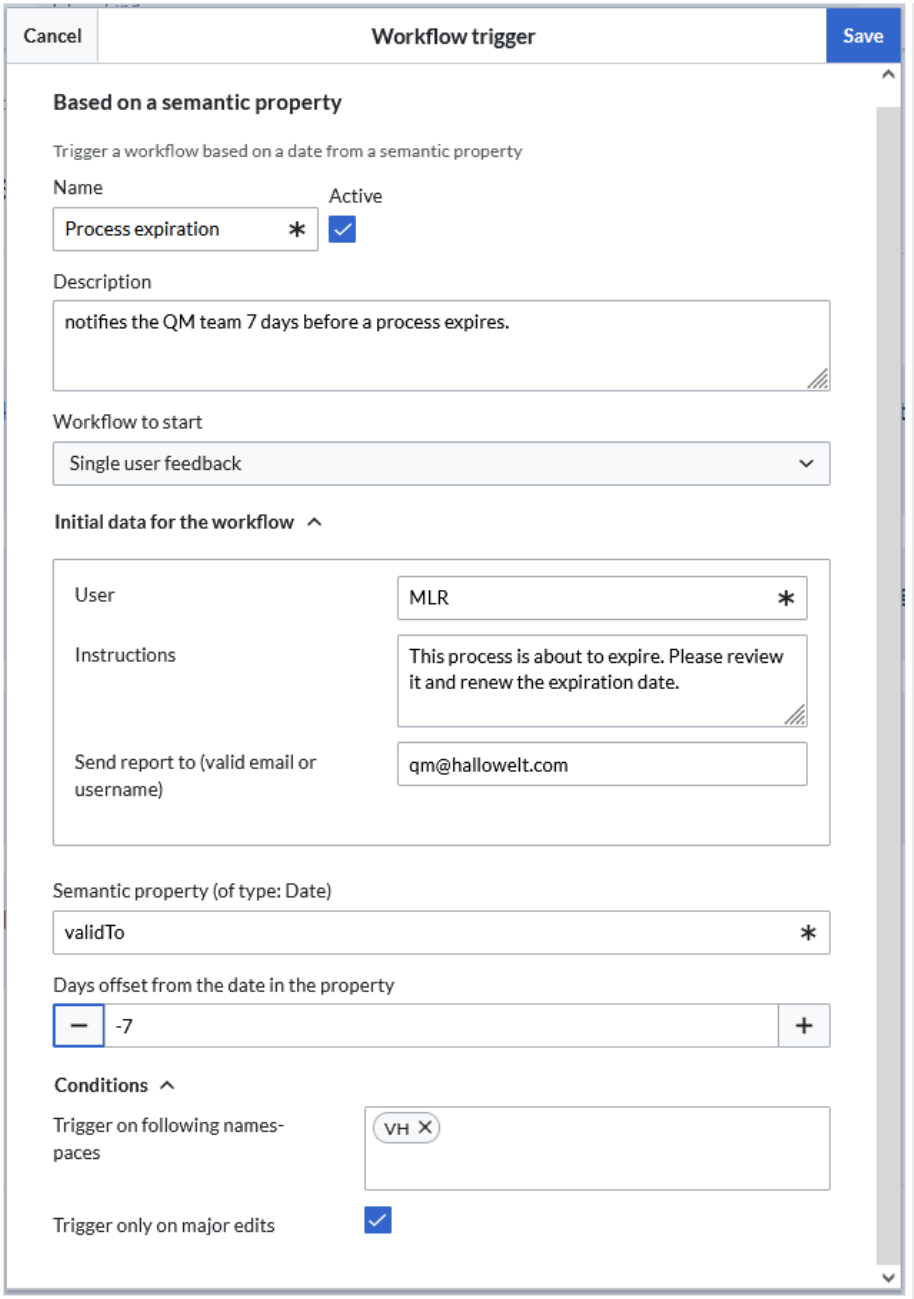
Types of triggers

All triggers share some commonalities:

Field	Description
Name	Give the trigger a unique name so that it is easy to identify its purpose.
Description	Describe to other users in more detail what this workflow does.
Workflow to start	Select the workflow that will be triggered.
Initial data for the workflow	You can provide some default data to be used in the workflow. In a manual workflow trigger, this data can later be overwritten when the workflow is started by a user.
Conditions	Select whether this workflow should only be available in particular namespaces. If no namespace is given, the workflow is available for the entire wiki. If the workflow is based on an edit-event, you can limit the workflow to be triggered only for major edits. Edits that a user marks as a minor edit when saving the page will not trigger the workflow.

Based on a semantic property

If the wiki uses semantic properties of type *date*, a workflow can start based on this date. It is possible to define a*n offset* from the date, so that the workflow can start before or after the date in the associated property has been reached.



Semantic date trigger

Additional fields

Field	Description
Semantic property (of type: Date)	Name of the date-property in the wiki that will trigger the workflow
Days offset from the date in the property	Set to a negative number if the workflow should start before the property date (e.g., -7) Set to a positive number if the worklfow should start after the property date (e.g., 5)

On edit

A workflow can also be triggered each time a page is edited. For this trigger it makes sense to set the conditions to individual namespaces and to trigger only on major edits. Users should be instructed that this workflow exists and that they should mark edits as minor when saving the page so that no unnecessary workflows are triggered.

On page creation

When new pages are created, a workflow can also be triggered.

Manual

Workflows can also be selected to start manually in particular namespaces. For example, it is possible to create a manual trigger for the *Expert Document Control* workflow to approve pages. This workflow can then be set to be offered only in a particular namespace and assign an approval task to a particular reviewer group.

Creating a custom workflow

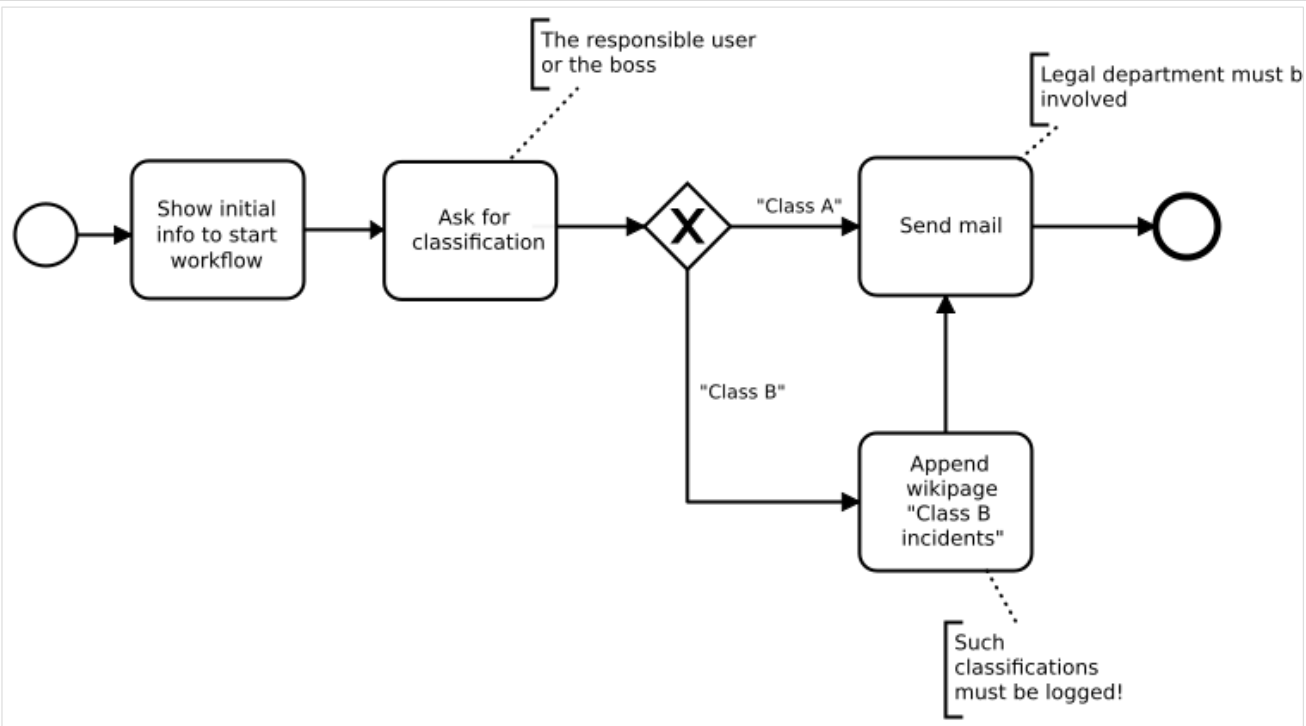
Contents

1 Defining the workflow 34
2 Steps 34
3 Instructions 35
3.1 Creating a custom workflow definition 35
3.1.1 Creating and connecting an initiation form 35
3.1.2 Creating and connecting the classification form 38
3.1.3 The gateway 39
3.1.4 Sending an mail to the legal department 40
3.1.5 Appending a wikipage 40
3.1.6 The final BPMN 41
4 Creating a workflow trigger 42
5 Testing the workflow 43
6 Using bpmn.io to create workflow definitions 43

Defining the workflow

Let's assume, we want to build a workflow, that asks for some classification of a wikipage. Based on the classification of the document, it either sends an e-mail to a member of the legal team or it first appends a wiki page and then sends the e-mail to the legal team.

The described workflow consists of four activities and one gateway. You can create a BPMN-Diagramm in the wiki to visualize the process:



Document classification workflow

Steps

To create the workflow, the following steps are necessary:

1. Creating a custom **workflow definition:** `MediaWiki:Classification-workflow.bpmn`
2. Creating and connecting an **initiation form** that adds some information about the workflow before it sends the task. `MediaWiki:ContentClassificationInit.form`
3. Creating and connecting the **classification form** that allows the assigned user to classify the document. `MediaWiki:ContentClassificationRequest.form`
4. Adding a **trigger** to the wiki that defines where and how to start the workflow.

Instructions

Creating a custom workflow definition

First, let's create a page called `MediaWiki:Classification-workflow.bpmn` with the most basic stub XML. Each workflow that you define in the wiki needs the elements that you see here:

- Line 1: The xml prolog
- Line 2: The definitions element, which denotes the namespaces where the workflow elements are defined.
- Line3: The process element, which contains all other elements.
- Line 5: The workflow runs in the context of a specific revision of a wiki page.
- Line 11: The workflow needs a startEvent and
- Line 18: The workflow needs an endEvent.

```
<?xml version="1.0" encoding="UTF-8"?>
<bpmn:definitions xmlns:bpmn="http://www.omg.org/spec/BPMN/20100524/MODEL" xmlns:wf="ht
tp://hallowelt.com/schema/bpmn/wf">
        <bpmn:process id="Classification_workflow_process">
            <bpmn:extensionElements>
                        <wf:context>
                                <wf:contextItem name="pageId"/>
                                <wf:contextItem name="revision"/>
                        </wf:context>
                </bpmn:extensionElements>

            <bpmn:startEvent id="TheStart">
                        <bpmn:outgoing>FromTheStartToInitializeWorkflow</bpmn:outgoing>
                </bpmn:startEvent>
                <bpmn:sequenceFlow id="FromTheStartToInitializeWorkflow" sourceRef="The
Start" targetRef="InitializeWorkflow" />

        <!-- ... -->

                <bpmn:endEvent id="TheEnd">
                        <bpmn:incoming>FromSendMailToTheEnd</bpmn:incoming>
                </bpmn:endEvent>

        </bpmn:process>

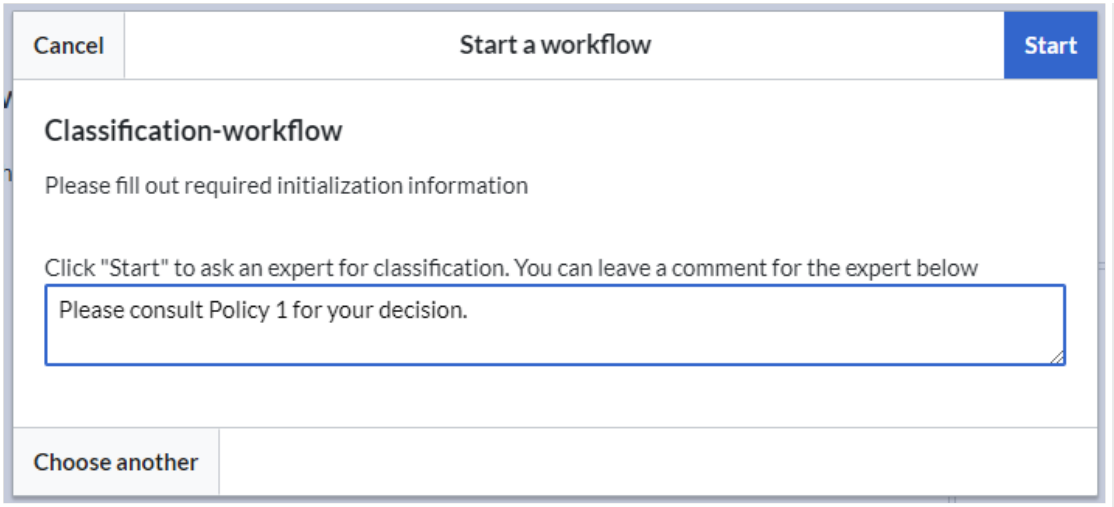
</bpmn:definitions>
```

The above stub xml only features the "Start" and "End" event including - yet to be defined - "outgoing" and "incoming" flow references.

Creating and connecting an initiation form

The initiation form allows to interact with the user who starts the workflow. It explains what happens when the workflow is started and allows to add a comment to provide some context for the user task.

We are using the MediaWiki namespace here to make sure not every user can edit the form later. But in general, such forms can be created everywhere in the wiki.



Worfklow initiation form

To create the initiation form:

1. **Create** the page `MediaWiki:ContentClassificationInit.form`
2. **Paste** the following form definition in source edit mode:

```
{
    "lang": "json",
    "form_name": "ContentClassificationInit",
    "items": [
        {
            "name": "intro",
            "widget_label": "Click  \"Start\" to ask an expert for classification.
You can leave a comment for the expert below",
            "type": "label"
        },
        {
            "name": "comment",
            "label": "Comment",
            "noLayout": true,
            "showOn": [
                "create",
                "edit",
                "view"
            ],
            "editableOn": [
                "create",
                "edit"
            ],
            "type": "textarea"
        }
    ]
}
```

3. **Save** the page.

Next, we tell the BPMN with the following userTask to show the form:

```
                <bpmn:userTask id="InitializeWorkflow" name="Start Content
Classification Workflow">
                        <bpmn:extensionElements>
                                <wf:type>custom_form</wf:type>
                                <wf:form>MediaWiki:ContentClassificationInit</wf:form>
                                <wf:initializer>true</wf:initializer>
                        </bpmn:extensionElements>
                        <bpmn:property name="comment"></bpmn:property>
                        <bpmn:incoming>FromTheStartToInitializeWorkflow</bpmn:incoming>
```

```
                        <bpmn:outgoing>FromInitializeWorkflowToAskForClassification<
/bpmn:outgoing>
                </bpmn:userTask>
                <bpmn:sequenceFlow id="FromInitializeWorkflowToAskForClassification" so
urceRef="InitializeWorkflow" targetRef="AskForClassification" />
```

Let's look at what this does:

- The `id` and `name` of this activity are set to "InitializeWorkflow" and "Start Content Classification Workflow". Both values do not need to match, but usually they do.
- <wf:type> is *custom_form* and tells the workflow that a form directly in the wiki (as opposed to a form located in code) is available.
- <wf:form> points to the actual form page in the wiki
- <wf:initializer> is set to *true* since it is used to show or collect some information before the actual workflow starts.

At the end, we have the incoming and outgoing flows referenced. (Note: the order of elements does not matter usually; only the nesting is important).

We add this **userTask** on line 16, after the line *<bpmn:sequenceFlow id="FromTheStartToInitializeWorkflow" sourceRef="TheStart" targetRef="InitializeWorkflow" />*:

```
<?xml version="1.0" encoding="UTF-8"?>
<bpmn:definitions xmlns:bpmn="http://www.omg.org/spec/BPMN/20100524/MODEL" xmlns:wf="ht
tp://hallowelt.com/schema/bpmn/wf">
        <bpmn:process id="Classification_workflow_process">
            <bpmn:extensionElements>
                        <wf:context>
                                <wf:contextItem name="pageId"/>
                                <wf:contextItem name="revision"/>
                        </wf:context>
                </bpmn:extensionElements>

            <bpmn:startEvent id="TheStart">
                <bpmn:outgoing>FromTheStartToInitializ</bpmn:outgoing>
        </bpmn:startEvent>
       <bpmn:sequenceFlow id="FromTheStartToInitializeWorkflow" sourceRef="TheStart" ta
rgetRef="InitializeWorkflow" />

                <bpmn:userTask id="InitializeWorkflow" name="Start Content
Classification Workflow">
                        <bpmn:extensionElements>
                                <wf:type>custom_form</wf:type>
                                <wf:form>MediaWiki:ContentClassificationInit</wf:form>
                                <wf:initializer>true</wf:initializer>
                        </bpmn:extensionElements>
                        <bpmn:property name="comment"></bpmn:property>
                        <bpmn:incoming>FromTheStartToInitializeWorkflow</bpmn:incoming>
                        <bpmn:outgoing>FromInitializeWorkflowToAskForClassification<
/bpmn:outgoing>
                </bpmn:userTask>
                <bpmn:sequenceFlow id="FromInitializeWorkflowToAskForClassification" so
urceRef="InitializeWorkflow" targetRef="AskForClassification" />

                <bpmn:endEvent id="TheEnd">
                        <bpmn:incoming>FromSendMailToTheEnd</bpmn:incoming>
                </bpmn:endEvent>

        </bpmn:process>

</bpmn:definitions>
```

Creating and connecting the classification form

The workflow will show a classification form to a user. This user will be defined in the workflow in a later step.

To create the classification form:

1. **Create** the page `MediaWiki:ContentClassificationRequest.form`
2. **Paste** the following form definition in source edit mode:

```
{
        "lang": "json",
        "form_name": "ContentClassificationRequest",
        "items": [
                {
                        "name": "intro",
                        "widget_label": "Please review the given document and select
an appropriate classification for the content from the list below",
                        "type": "label"
                },
                {
                        "name": "classification",
                        "label": "Classification",
                        "required": true,
                        "options": [
                                {
                                        "data": "CLSA",
                                        "label": "Class A"
                                },
                                {
                                        "data": "CLSB",
                                        "label": "Class B"
                                }
                        ],
                        "type": "dropdown",
                        "widget_$overlay": true
                }
        ]
}
```

Next, we tell the BPMN to show the form to user that receives the workflow task. For that, we add a *userTask* activity:

```
        ...
                <bpmn:userTask id="AskForClassification" name="Provide classification">
                        <bpmn:extensionElements>
                                <wf:type>custom_form</wf:type>
                                <wf:form>MediaWiki:ContentClassificationRequest</wf:
form>
                            <wf:initializer>true</wf:initializer>
                        </bpmn:extensionElements>
                        <bpmn:property name="assigned_user">
                                <![CDATA[{{ROOTPAGENAME:{{#show:{{FULLPAGENAME}}|?
Responsible|link=none|default=TheBoss}}}}]]>
                        </bpmn:property>
                        <bpmn:property name="due_date">
                                <![CDATA[{{#time:YmdHis|now + 7 days}}]]>
                        </bpmn:property>
                        <bpmn:property name="classification"></bpmn:property>
                        <bpmn:incoming>FromTheStartToAskForClassification</bpmn:
incoming>
                        <bpmn:outgoing>FromAskForClassificationToGateway</bpmn:
```

```
outgoing>
                </bpmn:userTask>
                <bpmn:sequenceFlow id="FromAskForClassificationToGateway" sourceRef="As
kForClassification" targetRef="Gateway" />
                ...
```

We should have a close look. This userTask has some additional properties besides the form reference (which was already explained in the initiation form):

- **assigned user:** The `assigned_user` property is mandatory, as the workflow engine must know which user to query. In this case, we use some wikitext magic to calculate the assigned user from the context. The value consists of a combination of a wikitext variable and a parserfunction (this particular parserfunction is defined by the Semantic MediaWiki extension, which can be seen as a dependency here). The `{{#show}}` parserfunction tries to get a username from a semantic property, that may or may not be set on the page the workflow is started on. If it does not find a proper information, it will fallback to `TheBoss` (assuming that such a user exists on the wiki). The `{{ROOTPAGENAME}}` variable is just an easy way to strip the "User" namespace, if the looked up value was something like `User:JaneDoe` instead of just `JaneDoe`.
- **due date:** The `due_date` property is also mandatory. All user facing activities need a due date. If the running activity is overdue, the workflow engine will end the workflow. In this case, we implement a concept of "lay days", as we do not have an absolute due date, but calculate it from the time the activity gets started using the `{{#time}}` parserfunction.
- **classification:** The `classification` property is a random one. It must be specified, in order to allow the form to set it and the workflow context to access it. We could have provided a default value, but we want to keep it empty.

At the end, we have the incoming and outgoing flows referenced. (Note: the order of elements does not matter usually; only the nesting is important).

The gateway

Now we can proceed with adding the gateway.

The gateway will provide the two necessary paths:

1. If the assigned user classified the document as Class A (CLSA), an email will be sent to the legal department.
2. If the assigned user classified the document as Class B (CLSB), a wiki page will be appended with the content defined in the task "AppendWikipage"

```
        ...
                <bpmn:exclusiveGateway id="Gateway" name="AskForClassification.
classification">
                        <bpmn:incoming>FromAskForClassificationToGateway</bpmn:
incoming>
                        <bpmn:outgoing>FromGatewayToSendMail</bpmn:outgoing>
                        <bpmn:outgoing>FromGatewayToAppendWikipage</bpmn:outgoing>
                </bpmn:exclusiveGateway>
                <bpmn:sequenceFlow id="FromGatewayToSendMail" name="CLSA" sourceRef="Ga
teway" targetRef="SendMail" />
                <bpmn:sequenceFlow id="FromGatewayToAppendWikipage" name="CLSB" sourceR
ef="Gateway" targetRef="AppendWikipage" />
                ...
```

After setting up the gateway, we need to define the tworelated activities.

Sending an mail to the legal department

For case 1, we need the workflow to immediately send an email:

```
        ...
                <bpmn:task id="SendMail" name="Send mail">
                        <bpmn:extensionElements>
                                <wf:type>send_mail</wf:type>
                        </bpmn:extensionElements>
                        <bpmn:property name="recipient">
                                <![CDATA[legal@company.local]]>
                        </bpmn:property>
                        <bpmn:property name="subject">
                                <![CDATA[New "{{{AskForClassification.
classification}}}" content: {{FULLPAGENAME}}]]>
                        </bpmn:property>
                        <bpmn:property name="body">
                                <![CDATA[Please check further actions now!]]>
                        </bpmn:property>
                        <bpmn:incoming>FromGatewayToSendMail</bpmn:incoming>
                        <bpmn:incoming>FromAppendWikipageToSendMail</bpmn:incoming>
                        <bpmn:outgoing>FromEndMailToTheEnd</bpmn:outgoing>
                </bpmn:task>
                <bpmn:sequenceFlow id="FromEndMailToTheEnd" sourceRef="SendMail" target
Ref="TheEnd" />
                ...
```

Appending a wikipage

For case 2, we want the workflow to append the existing page *Classification_incidents* with the text shown in the *content* property.

This requirement can be fulfilled with the *edit_page* activity type:

```
        ...
                <bpmn:task id="AppendWikipage" name="Append wikipage">
                        <bpmn:extensionElements>
                                <wf:type>edit_page</wf:type>
                        </bpmn:extensionElements>
                        <bpmn:property name="title" default="Classification_incidents"/
>
                        <bpmn:property name="user" default="MediaWiki default"/>
                        <bpmn:property name="content" default="* [[{{FULLPAGENAME}}]]
was classified {{{AskForClassification.classification}}}"/>
                        <bpmn:property name="mode" default="append"/>
                        <bpmn:property name="minor" default="1"/>
                        <bpmn:incoming>FromGatewayToAppendWikipage</bpmn:incoming>
                        <bpmn:outgoing>FromAppendWikipageToSendMail</bpmn:outgoing>
                </bpmn:task>
                <bpmn:sequenceFlow id="FromAppendWikipageToSendMail" sourceRef="AppendW
ikipage" targetRef="SendMail" />
                ...
```

The following properties are set for this activity type:

- *name:* title of the wiki page to which the content will be appended.
- *user:* user that is shown in the version history of the wiki page that was appended.
- *content:* text that is added to the wiki page.
- *mode:* shows where the text is added to the wiki page (append or ???)
- *minor: s*ets this page revision as a minor revision (1) or major revision (2).

After this task is completed, the workflow will send an email to the legal department.

The final BPMN

In the end, the bpmn page for the workflow looks like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<bpmn:definitions
        xmlns:bpmn="http://www.omg.org/spec/BPMN/20100524/MODEL"
        xmlns:wf="http://hallowelt.com/schema/bpmn/wf">

        <bpmn:process id="Classification_workflow_process">
                <bpmn:extensionElements>
                        <wf:context>
                                <wf:contextItem name="pageId"/>
                                <wf:contextItem name="revision"/>
                        </wf:context>
                </bpmn:extensionElements>
                <bpmn:startEvent id="TheStart">
                        <bpmn:outgoing>FromTheStartToInitializeWorkflow</bpmn:outgoing>
                </bpmn:startEvent>
                <bpmn:sequenceFlow id="FromTheStartToInitializeWorkflow" sourceRef="The
Start" targetRef="InitializeWorkflow" />

                <bpmn:userTask id="InitializeWorkflow" name="Start Content
Classification Workflow">
                        <bpmn:extensionElements>
                                <wf:type>custom_form</wf:type>
                                <wf:form>MediaWiki:ContentClassificationInit</wf:form>
                                <wf:initializer>true</wf:initializer>
                        </bpmn:extensionElements>
                        <bpmn:property name="comment"></bpmn:property>
                        <bpmn:incoming>FromTheStartToInitializeWorkflow</bpmn:incoming>
                        <bpmn:outgoing>FromInitializeWorkflowToAskForClassification<
/bpmn:outgoing>
                </bpmn:userTask>
                <bpmn:sequenceFlow id="FromInitializeWorkflowToAskForClassification" so
urceRef="InitializeWorkflow" targetRef="AskForClassification" />

                <bpmn:userTask id="AskForClassification" name="Provide classification">
                        <bpmn:extensionElements>
                                <wf:type>custom_form</wf:type>
                                <wf:form>MediaWiki:ContentClassificationRequest</wf:
form>
                        </bpmn:extensionElements>
                        <bpmn:property name="assigned_user">
                                <![CDATA[{{ROOTPAGENAME:{{#show:{{FULLPAGENAME}}|?
Responsible|link=none|default=TheBoss}}}}]]>
                        </bpmn:property>
                        <bpmn:property name="due_date">
                                <![CDATA[{{#time:YmdHis|now + 7 days}}]]>
                        </bpmn:property>
                        <bpmn:property name="classification"></bpmn:property>
                        <bpmn:incoming>FromInitializeWorkflowToAskForClassification<
/bpmn:incoming>
                        <bpmn:outgoing>FromAskForClassificationToGateway</bpmn:
outgoing>
                </bpmn:userTask>
                <bpmn:sequenceFlow id="FromAskForClassificationToGateway" sourceRef="As
kForClassification" targetRef="Gateway" />

                <bpmn:exclusiveGateway id="Gateway" name="AskForClassification.
classification">
                        <bpmn:incoming>FromAskForClassificationToGateway</bpmn:
incoming>
                        <bpmn:outgoing>FromGatewayToSendMail</bpmn:outgoing>
                        <bpmn:outgoing>FromGatewayToAppendWikipage</bpmn:outgoing>
                </bpmn:exclusiveGateway>
```

```
                <bpmn:sequenceFlow id="FromGatewayToSendMail" name="CLSA" sourceRef="Ga
teway" targetRef="SendMail" />
                <bpmn:sequenceFlow id="FromGatewayToAppendWikipage" name="CLSB" sourceR
ef="Gateway" targetRef="AppendWikipage" />

                <bpmn:task id="SendMail" name="Send mail">
                        <bpmn:extensionElements>
                                <wf:type>send_mail</wf:type>
                        </bpmn:extensionElements>
                        <bpmn:property name="recipient">
                                <![CDATA[legal@company.local]]>
                        </bpmn:property>
                        <bpmn:property name="subject">
                                <![CDATA[New "{{{AskForClassification.
classification}}}" content: {{FULLPAGENAME}}]]>
                        </bpmn:property>
                        <bpmn:property name="body">
                                <![CDATA[Please check further actions now!]]>
                        </bpmn:property>
                        <bpmn:incoming>FromGatewayToSendMail</bpmn:incoming>
                        <bpmn:incoming>FromAppendWikipageToSendMail</bpmn:incoming>
                        <bpmn:outgoing>FromEndMailToTheEnd</bpmn:outgoing>
                </bpmn:task>
                <bpmn:sequenceFlow id="FromEndMailToTheEnd" sourceRef="SendMail" target
Ref="TheEnd" />

                <bpmn:task id="AppendWikipage" name="Append wikipage">
                        <bpmn:extensionElements>
                                <wf:type>edit_page</wf:type>
                        </bpmn:extensionElements>
                        <bpmn:property name="title" default="Classification_incidents"/
>
                        <bpmn:property name="user" default="MediaWiki default"/>
                        <bpmn:property name="content" default="* [[{{FULLPAGENAME}}]]
was classified {{{AskForClassification.classification}}}"/>
                        <bpmn:property name="mode" default="append"/>
                        <bpmn:property name="minor" default="1"/>
                        <bpmn:incoming>FromGatewayToAppendWikipage</bpmn:incoming>
                        <bpmn:outgoing>FromAppendWikipageToSendMail</bpmn:outgoing>
                </bpmn:task>
                <bpmn:sequenceFlow id="FromAppendWikipageToSendMail" sourceRef="AppendW
ikipage" targetRef="SendMail" />

                <bpmn:endEvent id="TheEnd">
                        <bpmn:incoming>FromSendMailToTheEnd</bpmn:incoming>
                </bpmn:endEvent>

        </bpmn:process>

</bpmn:definitions>
```

workflows-activity-editpage-summary

Creating a workflow trigger

For the workflow to appear in the wiki, we need to define a trigger:

1. **Click** *Workflow triggers* in the Global actions menu.
2. **Click** *Add new trigger*.
3. **Select** the option *Manual* from the dropdown menu.
4. **Click** *Continue*.
5. **Define** the settings for the workflow trigger:
   - *Name:* Name that is displayed in the workflow selection menu.
   - *Description:* Explanation of the function and special features of this trigger.
   - *Workflow to start:* Workflow triggered by this trigger. In our case Classificiation-workflow.

- *Initial data for the workflow* (optional): - Standard comment suggestion.
- *Conditions* (optional)*:* In which namespaces the workflow is displayed for selection.

6. **Click** *Save*.

Your workflow is now ready to be tested.

Testing the workflow

The workflow is now available in the wiki. You should test whether the following functionality is available:

- The workflow is available to be started in the namespaces defined in the workflow trigger (if there are restrictions here).
- The workflow is triggered.
- The workflow is listed in the Worklfows Overview page.
- The assigned user received a task.
- The workflow completes correctly in case A and B.

Using bpmn.io to create workflow definitions

Such a diagram can be created with the free bpmn.io service. The resulting BPMN file needs to be modified, before it can actually be imported and used in the wiki.