

Contents

1. Manual:Extension/Workflows	2
2. Manual:Extension/Workflows/Activity/ApprovePage	11
3. Manual:Extension/Workflows/Activity/CustomForm	12
4. Manual:Extension/Workflows/Activity/EditRequest	15
5. Manual:Extension/Workflows/Activity/GroupVote	16
6. Manual:Extension/Workflows/Activity/SendMail	18
7. Manual:Extension/Workflows/Activity/UserVote	22
8. Manual:Extension/Workflows/Triggers	25
9. Manual:Extension/Workflows/Tutorial	28
10. Reference:BlueSpiceGroupManager	38
11. Reference:FlaggedRevs	40
12. Reference:UnifiedTaskOverview	41
13. Reference:Workflows	42

Workflows

Contents

1 Introduction 3

2 Workflow activities 3

2.1 Single user approval 3

2.2 Expert document control 4

2.3 Group feedback 6

2.4 Feedback 7

3 Overview page 8

3.1 Tasks overview 8

4 Notifications 8

4.1 Events that trigger notifications 8

4.2 Sending out notifications 9

5 Workflow triggers 10

6 How to add a custom workflow 10

7 Permissions 10

8 Example tutorial 11

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.

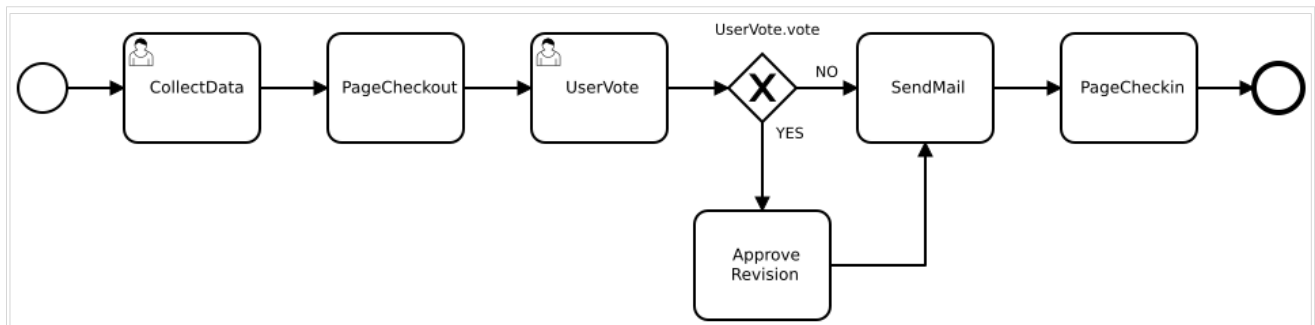
Workflow activities

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

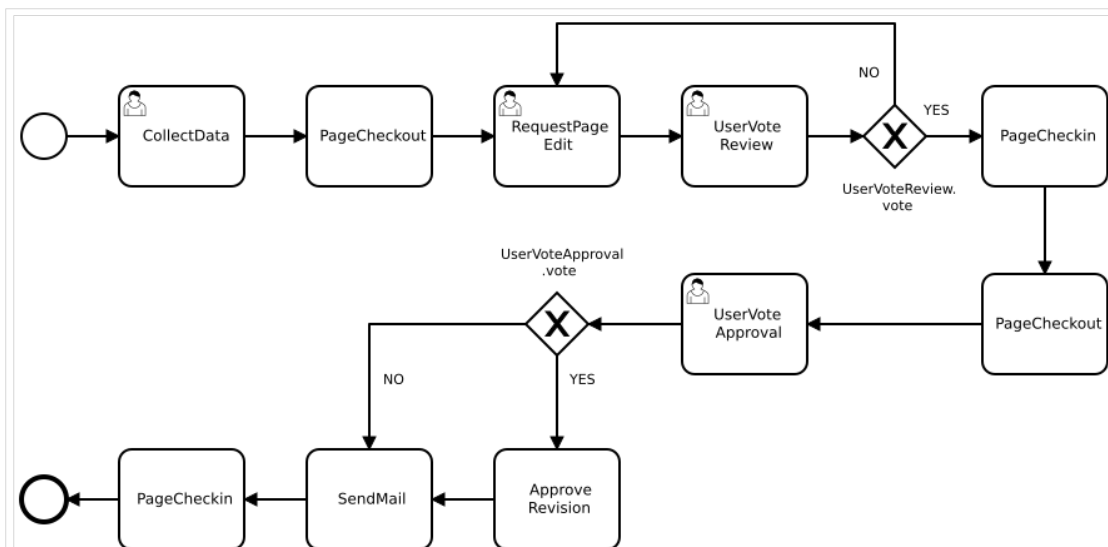
Activity	Description
CollectData	In the first workflow step, the workflow initiator enters the settings: <ul style="list-style-type: none">• <i>User</i>: ser who is assigned to the task.• <i>Instructions</i>: A comment or instructions for the user to understand the task.• <i>Send report to</i>: An email 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. The reviewer also cannot make any edits.
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). The workflow initiator gets an email about the voting result.
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.

[➞ View BPMN](#)

Expert document control

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

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



BPMN diagram of the "Expert document control" workflow

Activity	Description
CollectData	<p>In the first workflow step, the workflow initiator enters the settings:</p> <p><i>User:</i> User who is assigned to a task. Three different users have to be specified: Editor, Reviewer, Approver</p> <p><i>Instructions:</i> A comment or instructions for the users to understand their tasks.</p> <p><i>Send report to:</i> 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.</p>
PageCheckout	<p>The page is locked for users who do not participate in the workflow. Only the Editor (first workflow participant) can edit the page during checkout. While the Reviewer (second participant) of the workflow is reviewing the page, the page stays checked-out to thecan edit the page can edit the page during checkout.checkout. in case the Reviewer requests more edits.</p>
EditPage	<p>The Editor user can edit the page and completes the task without comment.</p>
UserVote	<p>After the Editor step has been completed, the Reviewer user can review the page and submit a vote. Editing by the Reviewer is not possible. 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.</p>

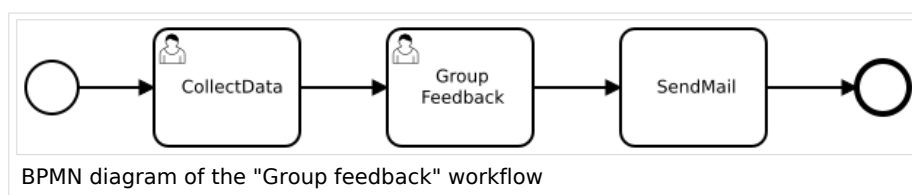
Activity	Description
	The workflow initiator gets an email about the voting result.
PageCheckin	After the Reviewer submits a positive vote (accept), the page is checked in and the workflow continues.
PageCheckout	In this step, the page checkout locks the page for editing completely. The Approver user 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.

→ [view BPMN](#)

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.



Activity	Description
	In the first workflow step, the workflow initiator enters the settings: <ul style="list-style-type: none">• <i>Group:</i> User group who is assigned to the task.• <i>Instructions:</i> A comment or instructions for the users to understand their task.

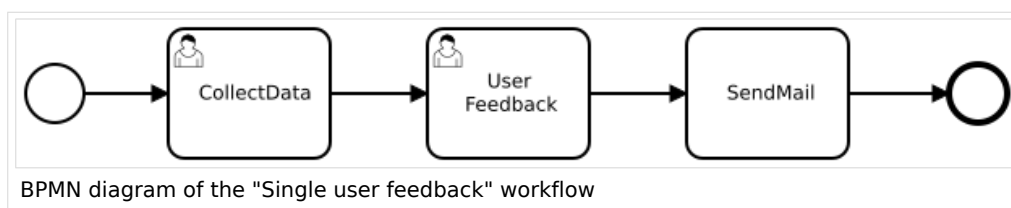
Activity	Description
CollectData	<ul style="list-style-type: none"><i>Send report to:</i> An email report with the results will be sent to this email address when the review is finished. If a username is specified here, an email address must be stored in the user administration so that the report can 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.

→ [view BPMN](#)

Feedback

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

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



Activity	Description
CollectData	In the first workflow step, the workflow initiator enters the settings: <ul style="list-style-type: none"><i>User:</i> User who is assigned to the task.<i>Instructions:</i> A comment or instructions for the users to understand their task.<i>Send report to:</i> 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.

→ [view BPMN](#)

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 related to this page						
<div>All Active</div>						
Type	Subject page	Current tasks	State	Started on	Last activity on	
Group Feedback	Main Page	-	Aborted	14:47, 29 November 2021	14:49, 29 November 2021	see details
Single user feedback	Main Page	-	Finished	15:43, 3 November 2021	15:43, 3 November 2021	see details
Expert document control	Main Page	UserVoteReview	Active	12:20, 21 December 2021	12:20, 21 December 2021	see details
Single user feedback	Main Page	-	Finished	16:34, 20 December 2021	17:13, 20 December 2021	see details
<div>< 1 of 3 ></div>						25 rows

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.

Notifications

Events that trigger notifications

There are two types of events that trigger notification

- **generic:** notifications happen for every workflow/activity type
- **activity-specific:** activities themselves can decide to send additional notifications

Triggering event	Recipients	Generic	Notes
Task started (task assigned)	All assigned users	Yes	Only triggered for type <i>UserInteractiveActivity</i> , i.e., only for activities that have users assigned.
UserVote	Initiator	Yes	The workflow initiator gets an email about the voting result.

Triggering event	Recipients	Generic	Notes
Workflow aborted (manual or automatic)	Initiator and all users that were assigned to the current task at time of aborting (not users who were assigned on previous tasks)	Yes	
Workflow ended (only when naturally ended, not when aborted)	Initiator	Yes	
DueDateClose (2 days before Workflow will expire)	Initiator and all currently assigned users	Yes	
Workflow expired	Initiator and all currently assigned users	Yes	Expiration is just a type of workflow abort, so the same notification as for abort will be sent with the reason explaining that the workflow expired.
Task delegated	User to whom the task was delegated	No	Specific to <i>UserVote</i> activity. After delegation, the newly assigned user will be considered assigned and will receive all further notifications that go out to assigned users.

Sending out notifications

- Users can choose whether to subscribe to e-mail notifications in their preferences. All users are force-subscribed to web notifications.
- Web notifications are sent out immediately after triggering, while email notifications will be sent async, on runJobs.php execution. This applies to notifications in general, not only to workflows

Workflow triggers

Workflows can either be started manually on each wiki page or started only under certain conditions using individual [workflow triggers](#). Triggers also allow to define in which namespaces both manual and automatic workflows are available.

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)

Permissions

The following permissions are used by this extension:

Permission	Included in role	Description
workflows-view	reader	<ul style="list-style-type: none">• allows viewing workflow elements, including listing of workflows (e.g., viewing all running workflows on a page)• user can view the page <i>Special:Workflows_overview</i>
workflows-execute	editor, reviewer, admin	<ul style="list-style-type: none">• allows starting a workflow and executing a task

Permission	Included in role	Description
workflows-admin	admin	<ul style="list-style-type: none">allows aborting, restoring and administering all workflowsuser fcdan view and edit the page <i>MediaWiki:WorkflowTriggers</i>

Example tutorial

You can follow our [tutorial for creating a custom workflow](#) that allows users to classify a document and notify a user about the classification.

➔ [Technical Reference: Workflows](#)

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 only be used with pages that have the approval feature ([FlaggedRevs](#)) enabled.

Short profile	
Name	ApprovePage
Async	Yes
BPMN type	<code>bpmn:Task</code>
BPMN Extension Element "wf:type"	<code>approve_revision</code>

Properties

Name of property	Description	Type
<code>comment</code>	The comment to be saved with the approval	string

Name of property	Description	Type
<div>revision</div>	The revision ID to approve. If not provided explicitly, the page revision of the workflow context will be used, which may be outdated.	int

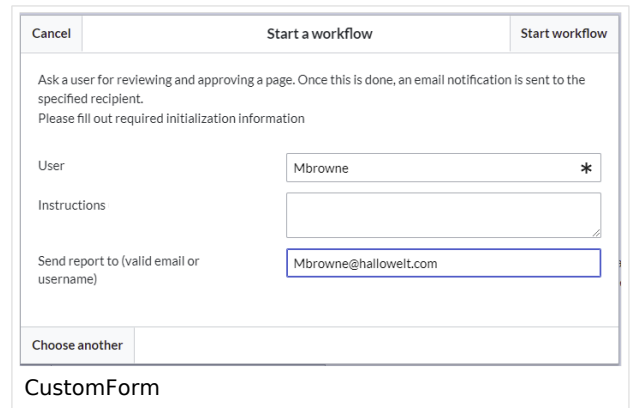
Manual:Extension/Workflows/Activity/CustomForm

Contents

1	Description	13
2	Extension elements	13
2.1	Example	14
3	Properties	14
3.1	Example	14

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.



Short profile	
Name	CustomForm
Async	No
Input/form	<i>Input depends on the workflow type, but in general:</i> <ul style="list-style-type: none">• 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
<code>due_date</code>	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" />
```

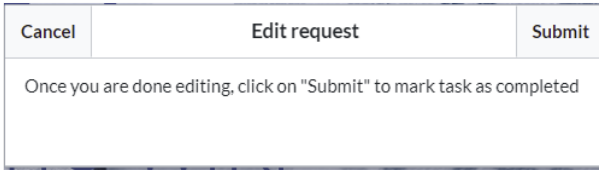
Given the activities `id` specified in the `<bpmn:userTask>` element was `"CollectData"`, connected activities would be able to access the output data in their respective properties like this:

```
<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/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	<code>bpmn:userTask</code>
BPMN Extension Element "wf:type"	<code>edit_request</code>

Properties

Name of property	Source	Description	Type
<code>due_date</code>	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
<code>assigned_user</code>	UIActivity	plain username (e.g. "WikiSysop") or user page ("User:WikiSysop"); Support for User-ID is not required	string
<code>instructions</code>	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 *GroupVote* is responsible for collecting necessary data about the voting of a group on a special topic. Unlike the [UserVote](#), it is not possible 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	<code>bpmn:userTask</code>
BPMN Extension Element "wf:type"	<code>group_vote</code>

Properties

Name of property	Description	Type
<code>due_date</code>	Due date for task completion	date /timestamp
<code>assigned_group</code>	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	<p>Not to be set in the workflow definition. Used to store data during the activities life cycle. E.g.</p> <pre>[{ "userName": "UserA", "vote": "yes", "comment": "Good" }, { "userName": "UserB", "vote": "no", "comment": "Not good" }]</pre> <p>Can be accessed by follow up activitites by e.g.</p> <pre>{{<GroupVoteActivityID>.0.userName}}</pre>	string
threshold_yes_unit	can be <code>user</code> or <code>percent</code> - 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 <code>user</code> or <code>percent</code> - 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="Definitions_lvrqlfw" 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>  
  </bpmn:process>  
</bpmn:definitions>
```

```
</bpmn:extensionElements>
<!-- StartEvent -->
<bpmn:startEvent id="TheStart">
  <bpmn:outgoing>FromTheStartToGroupVote</bpmn:outgoing>
</bpmn:startEvent>
<bpmn:sequenceFlow id="FromTheStartToGroupVote" sourceRef="TheStart" targetRef="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="GroupVote" 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" sourceRef="GatewayGroupVote" targetRef="TheEnd"/>
<bpmn:sequenceFlow id="FromGatewayGroupVoteToTheEndNo" name="no" sourceRef="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	20
2 Properties	21
2.1 Input	21
2.2 Output	21

3	Extension elements	22
---	--------------------	----

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 <ul style="list-style-type: none">the name of the user who casted the votea comment, which was left by the userthe voting result	string	display

Extension elements

Name of extension element	Description	Type
type		

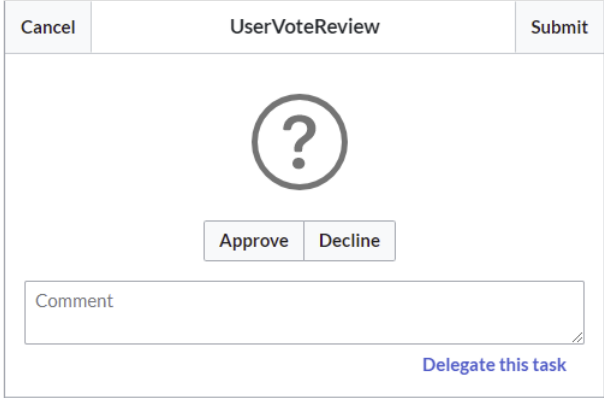
Manual:Extension/Workflows/Activity/UserVote

Contents

1	Description	23
2	Extension elements	24
3	Properties	24
3.1	Input	24
3.2	Output	25

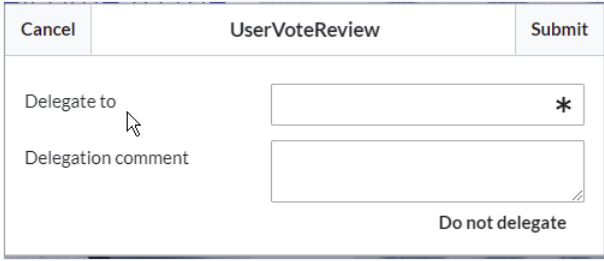
Description

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.



The dialog titled "UserVoteReview" has "Cancel" and "Submit" buttons. It features a large question mark icon, "Approve" and "Decline" buttons, a "Comment" text field, and a "Delegate this task" link.

Dialog for user voting



The dialog titled "UserVoteReview" has "Cancel" and "Submit" buttons. It includes a "Delegate to" label with a dropdown menu (marked with an asterisk), a "Delegation comment" text field, and a "Do not delegate" link.

Dialog for delegating

Short profile	
Name	UserVote
Async	No
Input/form	<ul style="list-style-type: none">• Submit / Cancel button• Voting button (approve/decline)• Textfield for comments• Link for delegating the task<ul style="list-style-type: none">○ Userpicker○ Textfield for comments
Associated to	<ul style="list-style-type: none">• 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	26
1.1	Based on a semantic property	26
1.2	On edit	28
1.3	On page creation	28
1.4	Manual	28

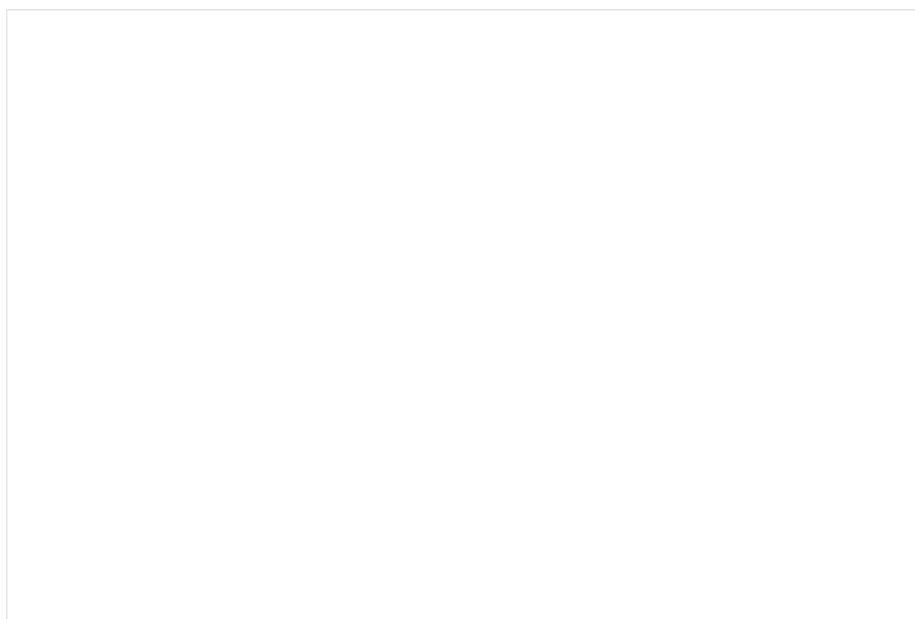
Types of triggers

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	<p>Select whether this workflow should only be available in particular namespaces. If no namespace is given, the workflow is available for the entire wiki.</p> <p>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.</p>

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 an *offset* from the date, so that the workflow can start before or after the date in the associated property has been reached.



Cancel
Workflow trigger
Save

Based on a semantic property

Trigger a workflow based on a date from a semantic property

Name Active

Process expiration * ☒

Description

notifies the QM team 7 days before a process expires.

Workflow to start

Single user feedback

Initial data for the workflow ^

User
MLR *

Instructions
This process is about to expire. Please review it and renew the expiration date.

Send report to (valid email or username)
qm@hallowelt.com

Semantic property (of type: Date)

validTo *

Days offset from the date in the property

- -7 +

Conditions ^

Trigger on following namespaces

VH X

Trigger only on major edits ☒

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 workflow 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

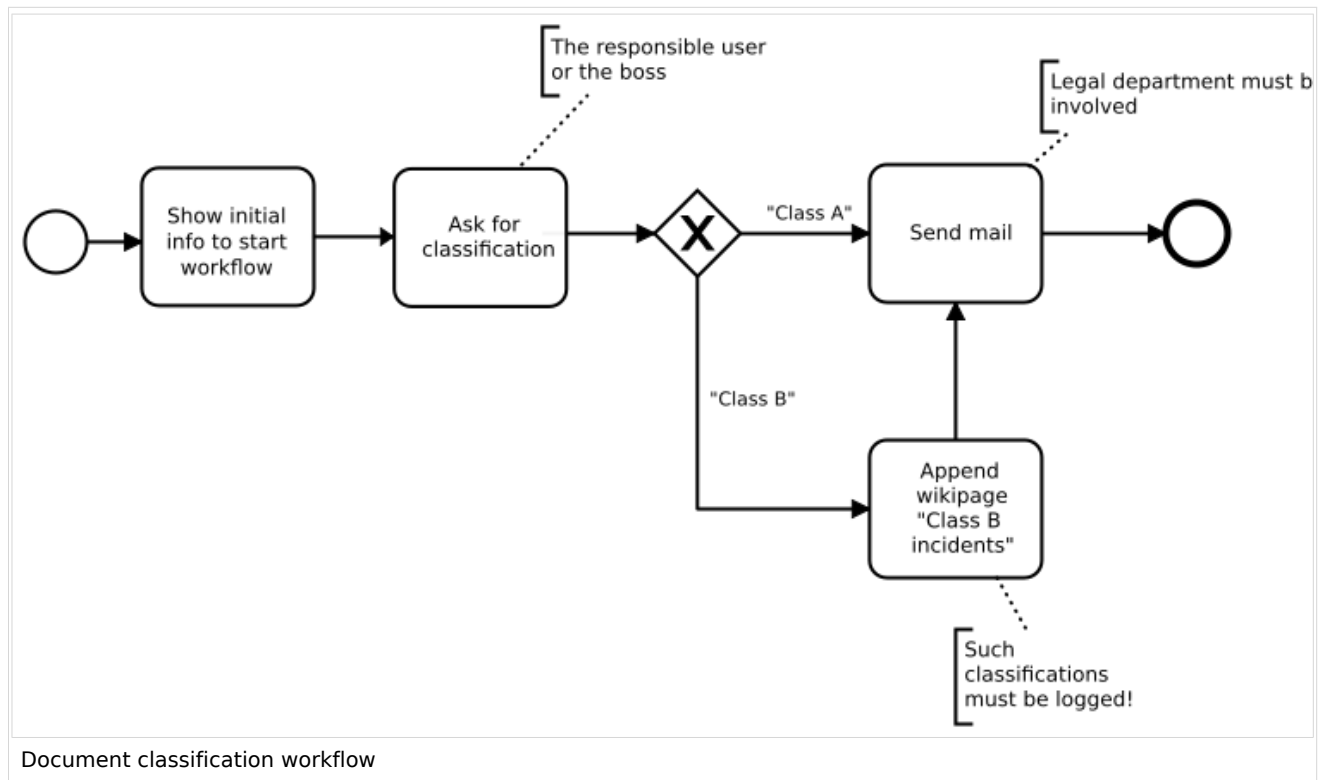
Contents

1	Defining the workflow	29
2	Steps	29
3	Instructions	30
3.1	Creating a custom workflow definition	30
3.1.1	Creating and connecting an initiation form	30
3.1.2	Creating and connecting the classification form	33
3.1.3	The gateway	34
3.1.4	Sending an mail to the legal department	35
3.1.5	Appending a wiki page	35
3.1.6	The final BPMN	36
4	Creating a workflow trigger	37
5	Testing the workflow	38
6	Using bpmn.io to create workflow definitions	38

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:



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="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="TheStart" 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.

Cancel Start a workflow Start

Classification-workflow

Please fill out required initialization information

Click "Start" to ask an expert for classification. You can leave a comment for the expert below

Please consult Policy 1 for your decision.

Choose another

Workflow 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="AskForClassification" 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 "AppendWikipedia"

```

    ...
    <bpmn:exclusiveGateway id="Gateway" name="AskForClassification.
classification">
        <bpmn:incoming>FromAskForClassificationToGateway</bpmn:
incoming>
        <bpmn:outgoing>FromGatewayToSendMail</bpmn:outgoing>
        <bpmn:outgoing>FromGatewayToAppendWikipedia</bpmn:outgoing>
        </bpmn:exclusiveGateway>
        <bpmn:sequenceFlow id="FromGatewayToSendMail" name="CLSA" sourceRef="Gateway" targetRef="SendMail" />
        <bpmn:sequenceFlow id="FromGatewayToAppendWikipedia" name="CLSB" sourceRef="Gateway" targetRef="AppendWikipedia" />
        ...

```

After setting up the gateway, we need to define the two related 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*: sets 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 Classification-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 Workflows 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.

Reference:BlueSpiceGroupManager

Extension: BlueSpiceGroupManager

→ [all extensions](#)

Overview			
Description:	Administration interface for adding, editing and deleting user groups and their rights		
State:	stable	Dependency:	BlueSpice
Developer:	HalloWelt	License:	GPL-3.0-only
Type:	BlueSpice	Category:	Administration

Overview			
Edition:	BlueSpice free, BlueSpice pro, BlueSpice Farm, BlueSpice Cloud	Version:	4.1+
? View user help page			

Features

GroupManager allows to create new groups, and to edit and delete existing groups.

Using the group concept, you can soften the wiki principle to work with a more granular permissions model. Using groups, administrators can assign different permissions to users. For example, a group can have read permissions, but no edit permissions. Use the group concept wisely, since otherwise the cooperative dynamic suffers.

With the Group manager, you can:

- create groups
- edit groups
- delete groups (system groups cannot be deleted or edited)

Users are assigned to groups in the **User manager**. Permissions are assigned to groups in the **Permission manager**.

Technical Information

This information applies to BlueSpice 4. Technical details for BlueSpice Cloud can differ in some cases.

Requirements

MediaWiki: 1.36.0
BlueSpiceFoundation: 4.1

Integrates into

Special pages

- GroupManager

Permissions

Name	Description	Role
groupmanager-viewspecialpage	Access to the special page Special:GroupManager	accountmanager, admin, maintenanceadmin

API Modules

- bs-groupmanager

Hooks

- [LoadExtensionSchemaUpdates](#)
- [MWStakeCommonUIRegisterSkinSlotComponents](#)
- [MWStakeDynamicConfigRegisterConfigs](#)

Accessibility

Test status:	2-testing complete
Checked for:	Web, Authoring tool
Last test date:	2022-08-05
WCAG level:	AA
WCAG support:	partially supports (workaround: no)
Comments:	<p>keyboard access works.</p> <p>Screenreader: Extjs: Pagination in grids not announcing button labels at top and pagination buttons at bottom; actions menu has no keyboard support.</p> <p>switching between browse/focus modes enables finding out how many pages there are and moving to the next page.</p> <p>erm:27716</p>
Extension type:	core
Extension focus:	admin

Reference:FlaggedRevs

Extension: FlaggedRevs

→ [all extensions](#)

Overview			
Description:	Allows for article content management by editors and reviewers.		
State:	stable	Dependency:	MediaWiki
Developer:	Aaron Schulz, Joerg Baach	License:	-
Type:	MediaWiki	Category:	Quality Assurance
Edition:	BlueSpice pro, BlueSpice Farm, BlueSpice Cloud	Version:	4.1- 4.2.x
? View help page			

Features

FlaggedRevs is a MediaWiki extension that activates an approval mechanism.

New versions of an article are initially marked as drafts and remain so until a user with revision rights checks and approves the page.

The MediaWiki extension is customized in BlueSpice. It uses the [FlaggedRevsConnector](#) to work alongside [workflows](#), realising the complete functionality of an editorial process with a final approval step.

Warning! The setting FR_INCLUDES_FREEZE for handling the state of included files of an approved wiki page has [no long term support by](#) and should not be introduced in a new wiki for that reason.

Reference:UnifiedTaskOverview

Extension: UnifiedTaskOverview

→ [all extensions](#)

Overview			
Description:	Provides the special page "My tasks"		
State:	stable	Dependency:	MediaWiki
Developer:	Hallo Welt!	License:	-
Type:	MediaWiki	Category:	Personalization
Edition:	BlueSpice pro, BlueSpice Farm, BlueSpice Cloud	Version:	4.1+

The new overview page "My Tasks" (Special:My_tasks) allows users to view their upcoming tasks in the wiki. The special page can be reached via the [user menu](#) under *Personal Tools* > *Tasks*.

- The task list shows [workflow tasks](#) and pending [read confirmations](#).
- Tasks can be directly [created on a page](#). v4.4+
- When new tasks exist, a red notification icon in the user menu alerts the user.
- Each task card leads to the wiki page that is linked to the task.

Special

My tasks

My tasks

Search

Main Page

User feedback

WikiSysop initiated this workflow

Instructions: Should we add more content?

OrgChart

User vote

WikiSysop initiated this workflow

Instructions: Please approve the page.

Hallo Welt! GmbH

Read confirmation required

QM:Introduction

Read confirmation required

Creating a Page

Read confirmation required

QM:Quality Management Representative

Read confirmation required

Special page "My tasks"

Reference:Workflows

Extension: Workflows

→ [all extensions](#)

Overview			
Description:	Allows to define and execute various workflows on the wiki		
State:	stable	Dependency:	MediaWiki
Developer:	Hallo Welt!	License:	GPL-3.0-only
Type:	MediaWiki	Category:	Quality Assurance
Edition:	BlueSpice pro, BlueSpice Farm, BlueSpice Cloud	Version:	4.1+
? View user help page			

Features

Workflow functionality based on BPMN. BlueSpice included some standard workflows. Custom workflows can be created additionally.

Technical Information

This information applies to BlueSpice 4. Technical details for BlueSpice Cloud can differ in some cases.

Requirements

MediaWiki: 1.39.0

Forms: *

- OOJSPlus: *

Integrates into

- BlueSpiceDiscovery
- UnifiedTaskOverview
- Workflows

Special pages

- [WorkflowsOverview](#)

Permissions

Name	Description	Role
workflows-admin	Manage workflows	admin
workflows-execute	Initiate and advance workflows	admin, reviewer, editor
workflows-view	View workflow items	reader

User options

Name	Value
echo-subscriptions-email-workflow-cat	1

Hooks

- [ArticleDeleteComplete](#)
- [BeforePageDisplay](#)
- [BlueSpiceDiscoveryTemplateDataProviderAfterInit](#)
- [CodeEditorGetPageLanguage](#)
- [ContentHandlerDefaultModelFor](#)
- [LoadExtensionSchemaUpdates](#)
- [MWStakeCommonUIRegisterSkinSlotComponents](#)
- [MWStakeRunJobsTriggerRegisterHandlers](#)
- [PageSaveComplete](#)
- [SetupAfterCache](#)
- [SkinTemplateNavigation::Universal](#)
- [UnifiedTaskOverviewGetTaskDescriptors](#)